FINAL MANAGEMENT REPORT

COOPERATIVE RESEARCH AGREEMENT

TELEMETRY DOWNLINK INTERFACES AND

LEVEL-ZERO PROCESSING

1 OCTOBER 1991

Prepared by
S. Horan J. Pfeiffer, and J. Taylor
Electrical and Computer Engineering
Dept. 3-O, Box 30001
Las Cruces, NM 88003-0001

(NASA-CR-188486) TELEMETRY DOWNLINK INTERFACES AND LEVEL-ZERO PROCESSING Final Report, 1 Oct. 1988 - 30 Sep. 1991 (New Mexico State Univ.) 144 p CSCL 09F G3/17

N92-18307

Unclas 0070335

TABLE OF CONTENTS

SECTION 1 - EXECUTIVE SUMMARY

This report is the Final Management Project Report as called for in the NASA Grant and Cooperative Agreement Handbook (NHB 5800.1A) for the cooperative research agreement titled "Research Relative to Telemetry Downlink Interfaces and Level-Zero Processing." This is NASA Cooperative Agreement number NCC 5-40.

This report covers the following topics

a) a description of the project task areas,

b) the project schedule,

c) the project status to date,

d) and other accomplishments relevant to the project's work.

This report covers the period from 1 October 1988 through 30 September 1991 and includes the contributions of all New Mexico State University faculty and students associated with the project.

SECTION 2 - PROJECT PERSONNEL

The research technical areas being investigated under the cooperative research agreement are as follows:

a) Processing of space-to-ground data frames,

b) Parallel architecture performance studies, and

c) Parallel programming techniques.

Additionally, there are the associated administrative areas of managing the grant which are

a) University administrative details and

b) Technical liaison between New Mexico State University and Goddard Space Flight Center.

There were management changes to the tasks through the life of the project. The final alignment of NMSU faculty members associated with each of these task areas are as follows:

a) Dr. Stephen Horan - team leader for the study of the processing of space-to-ground data frames and the project administrative and technical liaison,

b) Dr. Javin Taylor - team leader for the study of parallel architecture performance, and

c) Dr. Joseph Pfeiffer - team leader for the study of parallel programming techniques.

Additionally, Dr. Frank Carden was supported under this research activity during 1989.

To assist the NMSU faculty in the conduct of this research, the following students were employed as either graduate research assistants or as undergraduate research assistants at least part of the time during this agreement, as indicated:

a) Karim Al Hussini (graduate)

b) Edward Atel (undergraduate)

c) Georghios Georghiou (graduate),

d) Omar Hasan (graduate),

e) Mary Kennedy (graduate)

f) Brian Kopp (graduate)

g) Jaime Lara (graduate),

h) Michael Milyard (undergraduate),

i) Richard Oliver (graduate),

j) Krist Peterson (graduate),

k) John Polson (graduate),

l) Michael Ross (graduate),

m) O. Rich Smith (undergraduate),

n) James Thomas (graduate), and

o) John Watson (graduate).

SECTION 3 - PROJECT TASK AREAS

The technical work for this cooperative agreement is being divided into three sub-projects as follows:

I. Project 1: Processing Space-to-Ground Data Frames

Project Task Leader: Dr. Stephen Horan

Sub-project Areas

a. Study the functions required by the Space-to-Ground Transport (S/GT) and Level-Zero Processing (LZP) requirements.

b. Reconcile the CDC ETTS and Ford Aerospace simulations of S/GT and LZP requirements using available commercial simulation tools.

c. Study and evaluate the criteria, assumptions, and methodology defined in the CSC studies.

d. Investigate partitioning of the S/GT and LZP requirements. Determine partitioning alternatives, constraints, and resource requirements.

e. In concert with, or as an extension to, the CSC studies, develop representative simulation models of S/GT and LZP to evaluate the performance benefits of parallel processing.

II. Project 2: Parallel Architecture Performance Studies

Project Task Leader: Dr. Javin Taylor

Sub-Project Areas

a. Thoroughly study and understand the CDOS functions and matching of the CDOS functions to particular computer architectures, as described in the pertinent CSC documents.

b. Study and expand the CSC computer architecture classes and the matching of the CDOS functions to particular parallel computer architectures.

c. Validate the CSC reports with respect to parallel processing as it applies to the CDOS functions, especially the DHC mass-storage functions. Validate the CSC comparisons of parallel processing with serial processing. Study extending the CSC results to the Connection machine and neural networks.

d. Simulate pertinent CSC architecture/CDOS function matches using commercial simulation tools.

e. Evaluate local parallel-processing architecture with respect to the CSC computer architecture classes and the matching of the CDOS functions to particular architectures. Develop computer simulation models for this architecture using commercially-based simulation tools.

III. Project 3: Parallel Programming Techniques

Project Task Leader: Dr. Joseph Pfeiffer

Sub-project Areas

a. Select algorithms and develop benchmarks for low-level and high-level language evaluation. Develop benchmarks for operating system evaluation.

b. Survey and evaluate parallel processor operating systems including UNIX, MACH, and proprietary operating systems.

c. Evaluate ADA support in parallel implementation of benchmarking algorithms.

d. Survey and evaluate high-level language and development tools availability for candidate vendors selected in b. Consider source-level debuggers and programming support for Ada.

e. Survey and evaluate low-level language and development tools availability for candidate vendors selected in b. Consider machine-level debuggers.

4 - OTHER ACCOMPLISHMENTS

A. DEGREES AWARDED

From the list of graduate students, Mary Kennedy and John Polson have received their MS degrees using this research as their projects. John Watson is using this research and is expected to finish his PhD in 1992. Krist Peterson and John Polson have used the support of this project to pursue their PhD studies although their doctoral research has not been completed.

B. PAPERS PRESENTED

As submitted previously, papers dealing with this research have been presented as follows:

a) Stephen Horan made presentations at the 1989 International Telemetering Conference and the CACI Simulation Conference,

b) John Watson John Polson, and Dr. Pfeifer made presentations at the 1990 International Telemetering Conference,

c) Drs. Taylor and Horan presented the development of the simulation laboratory resources, partially funded by this effort, at the 1990 ASEE Conference.

5 - ADMINISTRATIVE PROBLEMS

We identified several problems with the accomplishment of the research tasks undertaken in this problem. The primary one was due to the distance between Las Cruces and GSFC. This prevented timely interaction between GSFC and the NMSU personnel. This was most telling when requirements for the CDOS would change or the CCSDS standards would be updated. At times, we were operating one year behind the latest changes. This meant that assumptions and methods were always changing and lagging behind current thinking. This could be changed on similar projects in the future by making the objectives more abstract and independent of work that may come under procurement confidentiality restrictions.

By the nature of the NASA communications structure and CCSDS protocols, they are not familiar to students brought on-board as research assistants and to most faculty. Much of the time working with students was spent educating them on the ways in which NASA communications operate. More time in the schedule was needed to perform this education process.

The combination of both effects did generate several schedule slips which made the research effort less effective than could have been. We believe that significant results were obtained, although, perhaps not those initially sought.

FINAL TECHNICAL SUMMARIES

1. DATA TRANSMISSION REQUIREMENTS

by

Stephen Horan, PhD
John C. Watson

Department of Electrical and Computer Engineering
New Mexico State University
Box 30001, Dept. 3-O
Las Cruces, NM 88003-0001

ABSTRACT

The study of data transport requirements has been conducted to understand the protocol requirements, transmission options, and methodology for predicting performance. In this report, we review previous simulation methodology and results and contrast that with the improved methodology developed here. That methodology is then applied to the transport of data in a space-station-like environment. From a review of CCSDS protocol requirements, we also review some areas for further research.

1.1 SIMULATION METHODOLOGY

Many different approaches can exist to study the transport of data from space payloads to the ground for processing. In this research effort, we have concentrated on the application of commercial tools to the problem to assess the prospects for transporting Space Station Freedom types of traffic loads to the ground via the Space Network. In this area, we have attempted to re-create some of the previous studies to validate our models and to better understand some of the history of previous projects. In this effort, we have looked at the CDC and Ford Aerospace studies. We will then present the methodology we believe to be superior to those studies to give (1) a more flexible means of simulating the process and (2) a more correct view of the data transport process. We will also give our review of some of the tools that are available for simulation methods.

1.1.1 PREVIOUS SIMULATION STUDIES

The two major efforts relevant to this research work were conducted by CDC and Ford Aerospace. In this section we briefly review those efforts.

CDC STUDY

Control Data Corporation (CDC) conducted a study to explore the benefits of packet

processing technology for space data transport. Their study was basically built around their Cyber computer architecture. They conducted simulation studies of the performance of a packet pointer processing architecture for processing space data. Mary Kennedy of New Mexico State University, in her Master's Technical Report, conducted a shadow simulation of the CDC project using the Network II.5 simulator. While Kennedy was able to validate the basic approach used by CDC, she found some serious problems. For example, the simulator used differing levels of resolution in each of its two parts. While this is, in general, a reasonable approach, especially when high-level languages are being used, the two parts used some differing assumptions as to what was exactly happening. Kennedy was unable to verify several assumptions on processing speeds and ordering.

FORD AEROSPACE STUDY

After the CDC study, Ford Aerospace performed a more general simulation with their FAST simulator. FAST is a proprietary tool written in ADA and runs on a VAX workstation. FAST does have the potential to modify the input data set for varying traffic loads. However, problems with FAST from our point of view include the following:
a) the means to change system parameters are not simple or obvious,
b) the data traffic is composed of one long data set for each traffic source,
c) the traffic really does not appear to have any traffic statistical dynamics included.
This last point is the most troublesome since packet statistical dynamics contain all of the "interesting" features of the data transport.

1.1.2 IMPROVED METHODOLOGY

As an improvement to the state of the data transport simulation art, we attempted to be more realistic and more user-friendly about the methodology for simulating the data transport process. This methodology includes the use of (1) commercial simulators, (2) high-level simulation languages, (3) linear programming. The commercial simulators were thought to provide the following advantages:

a) ease of configuration for varying scenarios,
b) standard tools for modeling generic components (processors, transmission buses, software processes, etc.),
c) easy validation of the model because we would be using pre-developed building blocks.

The use of linear programming techniques was included to speed up the optimization process. Commercial simulators can allow iteration through parameters, however, that can be a long process if the iterations do not start near an optimal configuration. We also decided to use a high-level language simulation tool to the suite to allow for cross-checking of results.

BASIC METHOD AND TOOLS

The commercial simulation tools that were chosen to work with were

a) Network II.5 - a Simscript-derivative marketed by CACI and available for use on VAX, SUN, and PC platforms, and
b) BONeS - a tool marketed by COMDISCO for use on a SUN platform.

The high-level simulation language chosen to work with was GPSS which was available for use on a mainframe. The linear programming model was LPDEMO also available for us on the PC.

The methodology for performing the simulation was to

a) parameterize the traffic model from the CDOS Traffic Model (CDOS 0915.0002 V3.1) into a series of statistical drivers partitioned by virtual channel,
b) encode the CCSDS data transport protocol for the space-segment into a series of constraint equations for the linear programming model,
c) encode the FDDI LAN parameters into a series of constrain equations for the linear programming model,
d) find the optimal configuration for LAN and packet parameters from the linear programming model by optimizing over several transport frames,
e) use Network II.5 and GPSS models of the same traffic, protocol, and LAN configuration to see how the optimization works with full statistical variation and determine resource implications.

From this methodology, suite of simulation results is being developed that will define a parameter space for optimum data transport parameters.

COMPARISON OF COMMERCIAL SIMULATOR TOOLS

The exercise of the simulators for this research tested the capabilities and usefulness of the commercial simulators. The paper "Validation of Priority FDDI LAN Simulators" attached to this report contains the results full testing of the simulators using the priority FDDI token LAN as a test article and then comparing the models with theoretical results. The results can be summarized as follows in Table 1. In defense of Network II.5, with CACI's help, we were eventually able to remove the disagreements with theory in simulating the priority token ring simulations. However, this was by introducing simulation constructs that either should have been part of the native package or been explicitly documented in the user's manual. We seriously doubt that a naive user would come up with this construct on their own. Hence, we still keep our overall rating of Network II.5 lower than it would have been if this were not a problem.

Table 1. Comparison of Experience with FDDI Simulation Models			
Attribute	**GPSS**	**BONeS**	**Network II.5**
Model Development	extensive hand coding	extensive combination of model primitives	fixed input parameters from a fixed menu of options
Automatic Model Iterations	not available	input parameter	not available
Level of Parameter Detail	as extensive as user codes	extensive	moderate
Ease of Modification	base code needs to be edited	graphical block diagram based user interface with user input of parameters	structured user interface with menus
Relative Execution Speed	fast	slow	moderate
Execution Platforms	mainframe	SUN-4	SUN-4, VAX workstation, PC
Single Asynchronous Priority Simulations	excellent agreement with theory	excellent agreement with theory	disagreement with theory
Multiple Asynchronous Priority Simulations	excellent agreement with theory	excellent agreement with theory	severe disagreement with theory

1.2 SIMULATION MODEL

The simulation model developed here is based on the CCSDS Principal Network Structure illustrated in Figure 1. This is based on the CCSDS 701.0-B-1 and related protocol standards. This structure is amplified by the assumption of the on-board communications architecture consisting of an FDDI LAN structure as given by J.E. Smith, D. Willett, and S. Paul, "Overview of the Space Station Communications Networks," *IEEE Networks Magazine*, Sept. 1990. This is illustrated in Figure 2. The details of the FDDI token ring come from the applicable industry standards. The data traffic was generated through a synthesis of the CDOS Traffic Model Version 3.1.

From these sources, the first step was to generate a statistical traffic model to cut down the number of drivers, and thereby increase simulation effectiveness, for the model. The results of the consolidation of the traffic is as shown in Tables 2 and 3.

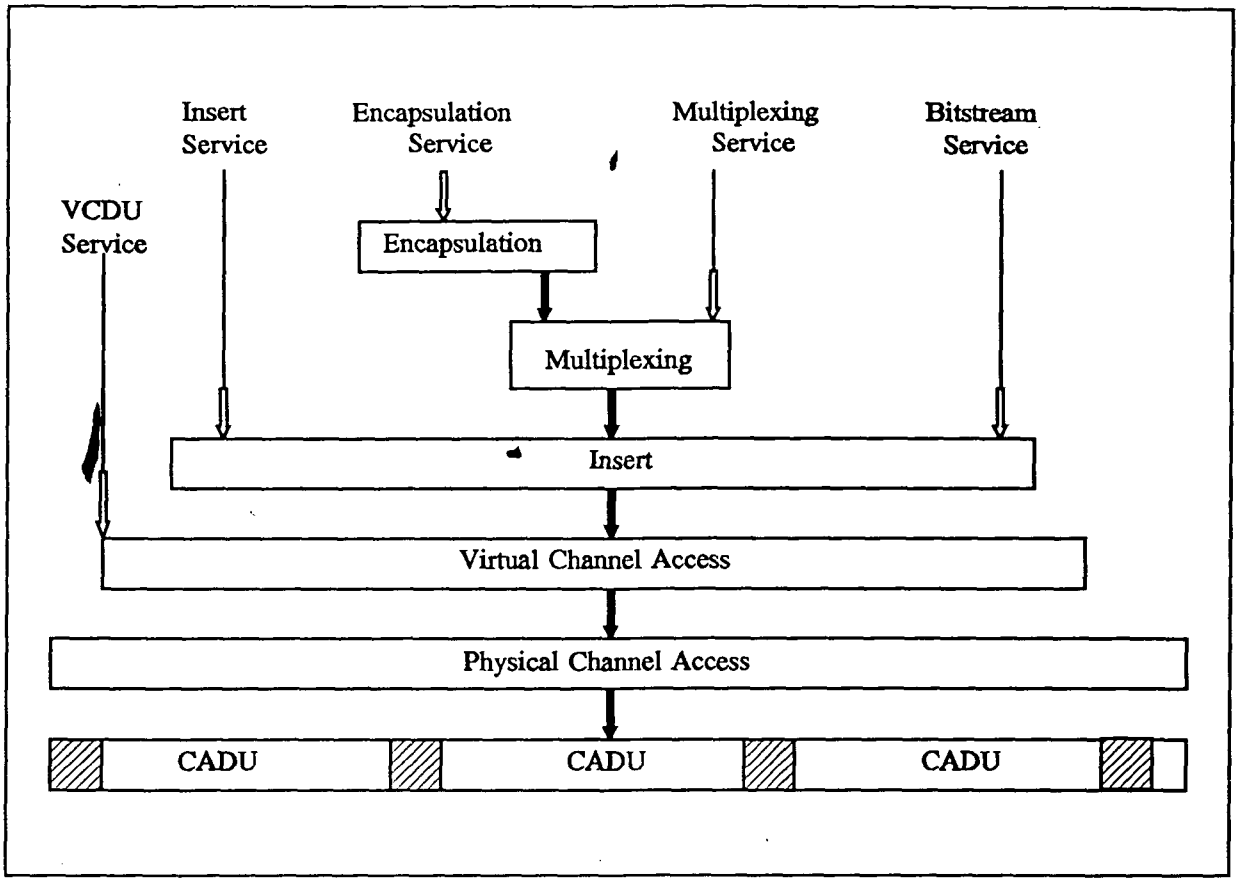


Figure 1 - CCSDS Principal Network Structure.

Table 2. Low Data Rate Aggregate Results		
Upper Limit (Kbps)	**Percentage of Total**	**Cumulative Percentage**
5.0	0.28	0.28
10.0	0.63	0.91
15.0	15.86	16.77
20.0	17.22	33.99
25.0	45.32	79.31
30.0	18.45	97.76
35.0	1.87	99.63
40.0	0.31	99.94
45.0	0.06	100

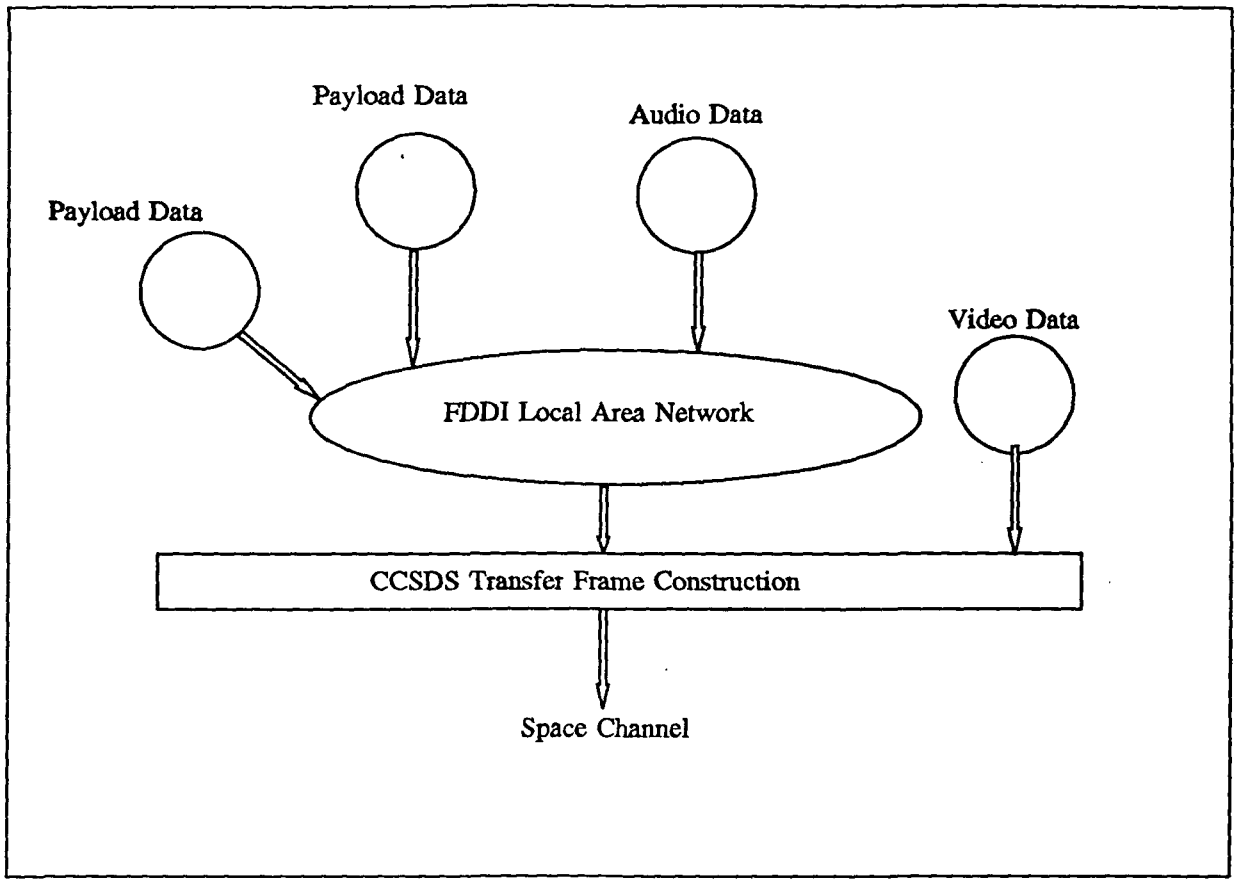


Figure 2 - Space Station LAN Structure.

Table 3. High Data Rate Aggregate Results		
Upper Limits (Mbps)	**Percentage of Total**	**Cumulative Percentage**
30.0	80.01	80.01
60.0	2.43	82.44
90.0	6.95	89.40
120.0	6.29	95.69
150.0	2.05	97.74
180.0	1.82	99.56
210.0	0.30	99.86
240.0	0.03	99.89
270.0	0.11	100

Next, a linear program for optimal results was developed. This model included as constraints

a) the size of PDUs for the various services and protocol levels,
b) the FDDI frame PDU size,
c) the FDDI timing,
d) the CADU timing on the Space Network,
e) assumed locations of buffers and buffer sizes, and
f) traffic model source amounts, including video data.

From this constraint list, optimal parameters were determined by optimizing not over one CADU frame but over many. The optimization criterion was minimizing the amount of fill data sent during each CADU. From this optimization, data transport throughput results were obtained as given in Figure 3.

Next, these models are configured as Network II.5 models and GPSS models. Both methods were chosen to
a) provide cross-checks on results and
b) provide information on relative ease of making the models.

The preliminary results are illustrated in Figures 4 and 5. These results are preliminary and the investigation is continuing to produce the full parameter space. The expected outputs of the simulation models include information on the following parameters

a) required on-board buffer space,
b) channel usage efficiency (ratio of fill to actual data),
c) reaction to traffic transients, and
d) data arrival timing.

The basic results to-date indicate that the CCSDS protocol standards can play effectively with the FDDI data transport on-board and with the Space Network channel.

1.4 EXTENSIONS FOR FURTHER WORK

The studies here can be used as a base methodology for further investigations of the space data transport areas. In particular, studies can be extended to the following areas:

a) utilizing the priority levels in the FDDI to affect data transport from different virtual channels; the present scheme treats all virtual channels with the same priority,
b) looking at the effects of lossless data compression on the transport; this would lessen the amount of data to be transported and affect the ground-segment processing for packet switching, and
c) CCSDS protocol testing to fully verify end-to-end protocol operations to convert the standards from paper artifacts to actual, working models.

Figure 3 Linear Optimization Optimal Results for Minimizing Transmitted Fill Data.

LP Model Transfer Frame Fill Data Rate

(Single FDDI, 68 msec)

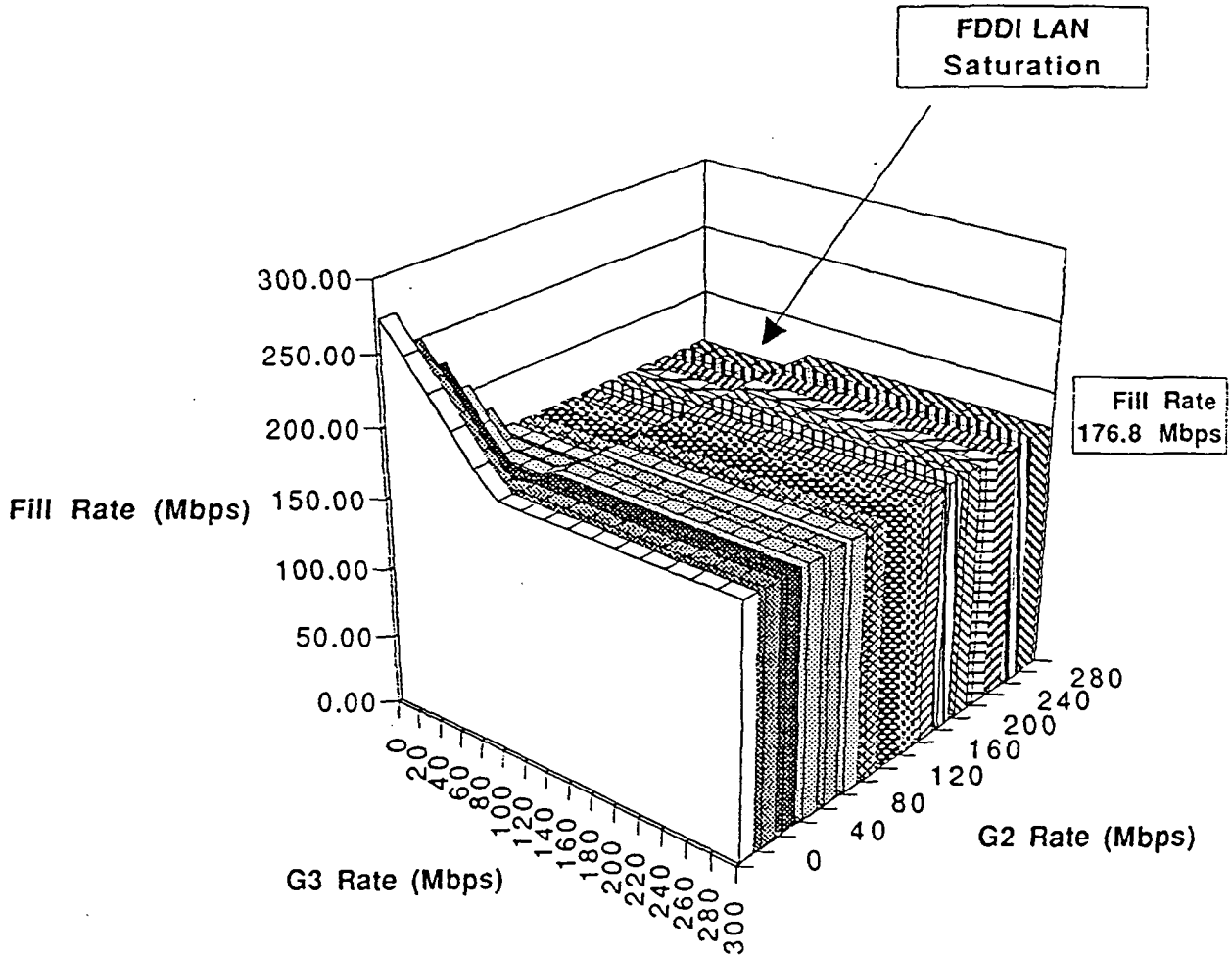
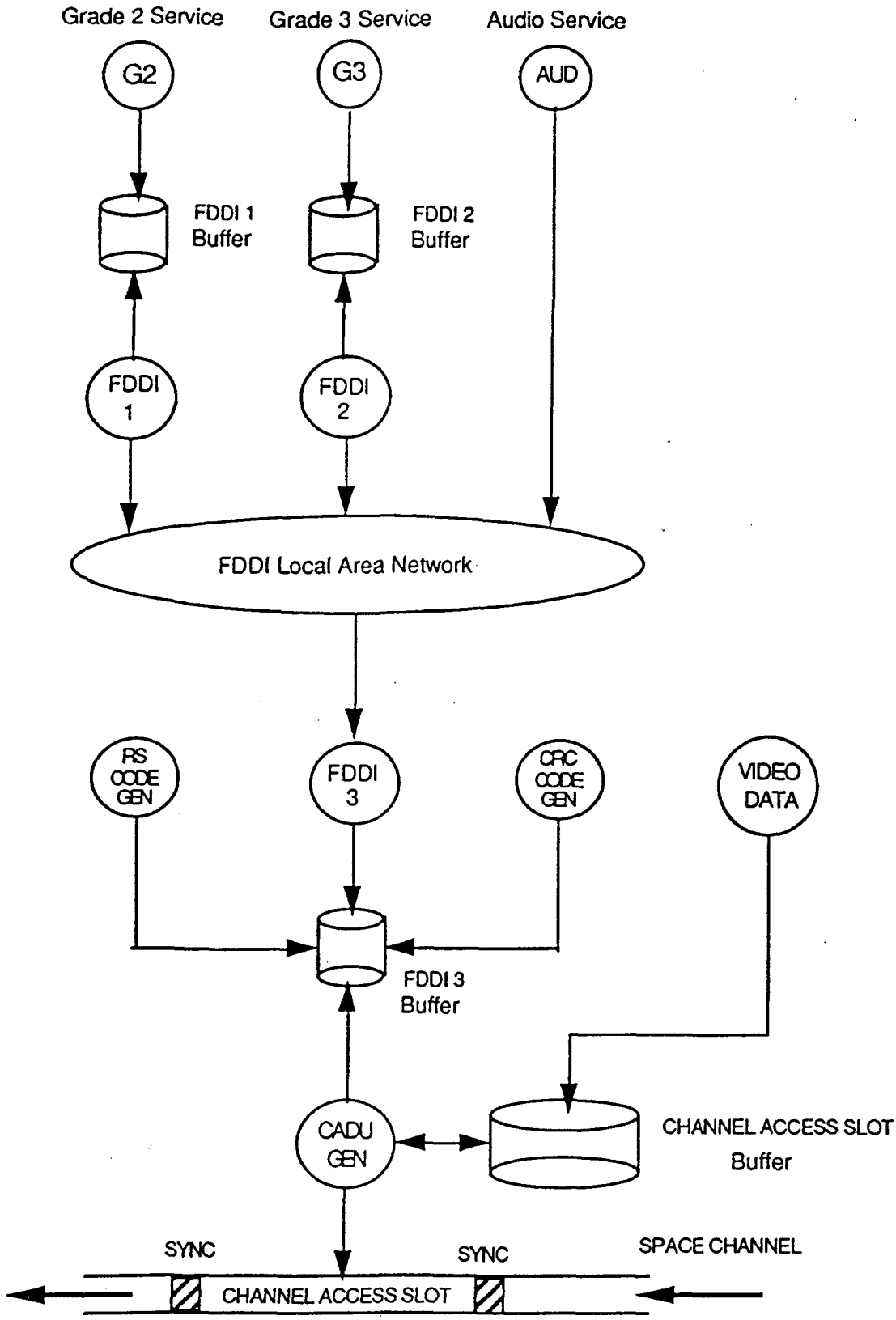


Figure 3. Single FDDI Baseline Configuration Fill Data

Figure 4 FDDI LAN Configuration and Simulated Fill Minimization and Buffer Requirements



Single FDDI LAN Space Data System

GPSS Model Transfer Frame Fill Data Rate

(Single FDDI, 68 msec)

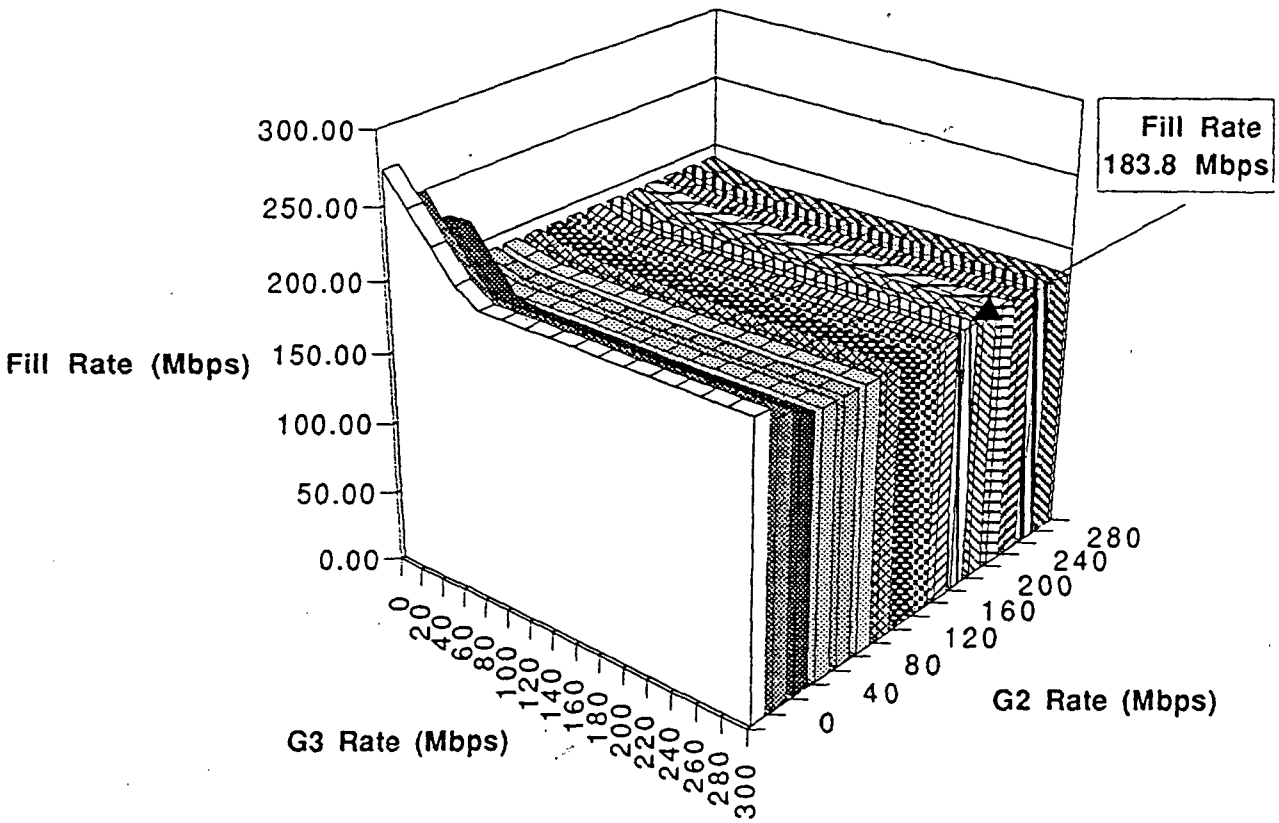


Figure 8.1 Single FDDI Baseline Configuration Fill Data

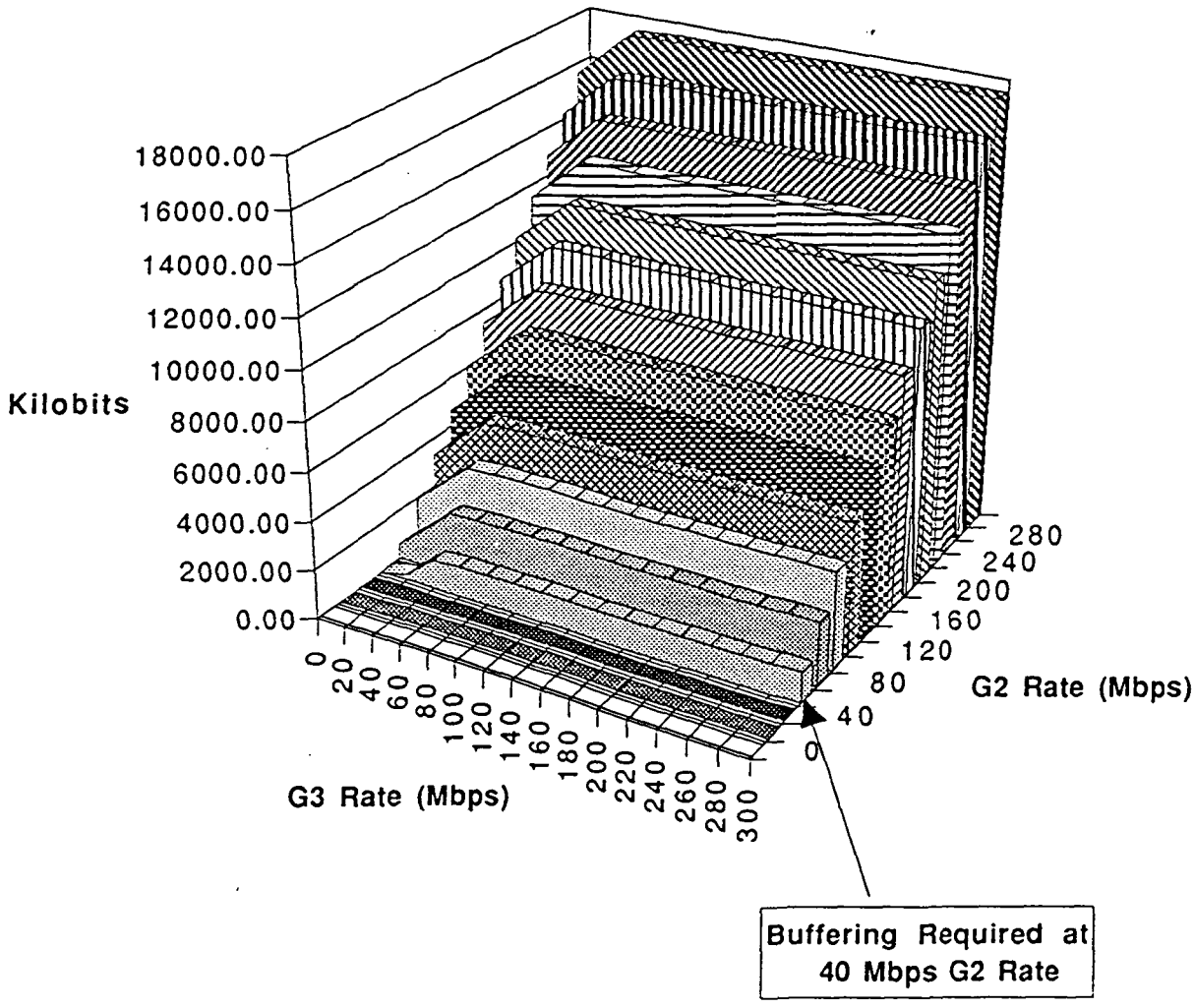


Figure 8.2 Single FDDI Baseline Configuration Grade 2 Buffer Capacity

GPSS Model Grade 3 Buffer Capacity

(Single FDDI, 68 msec)

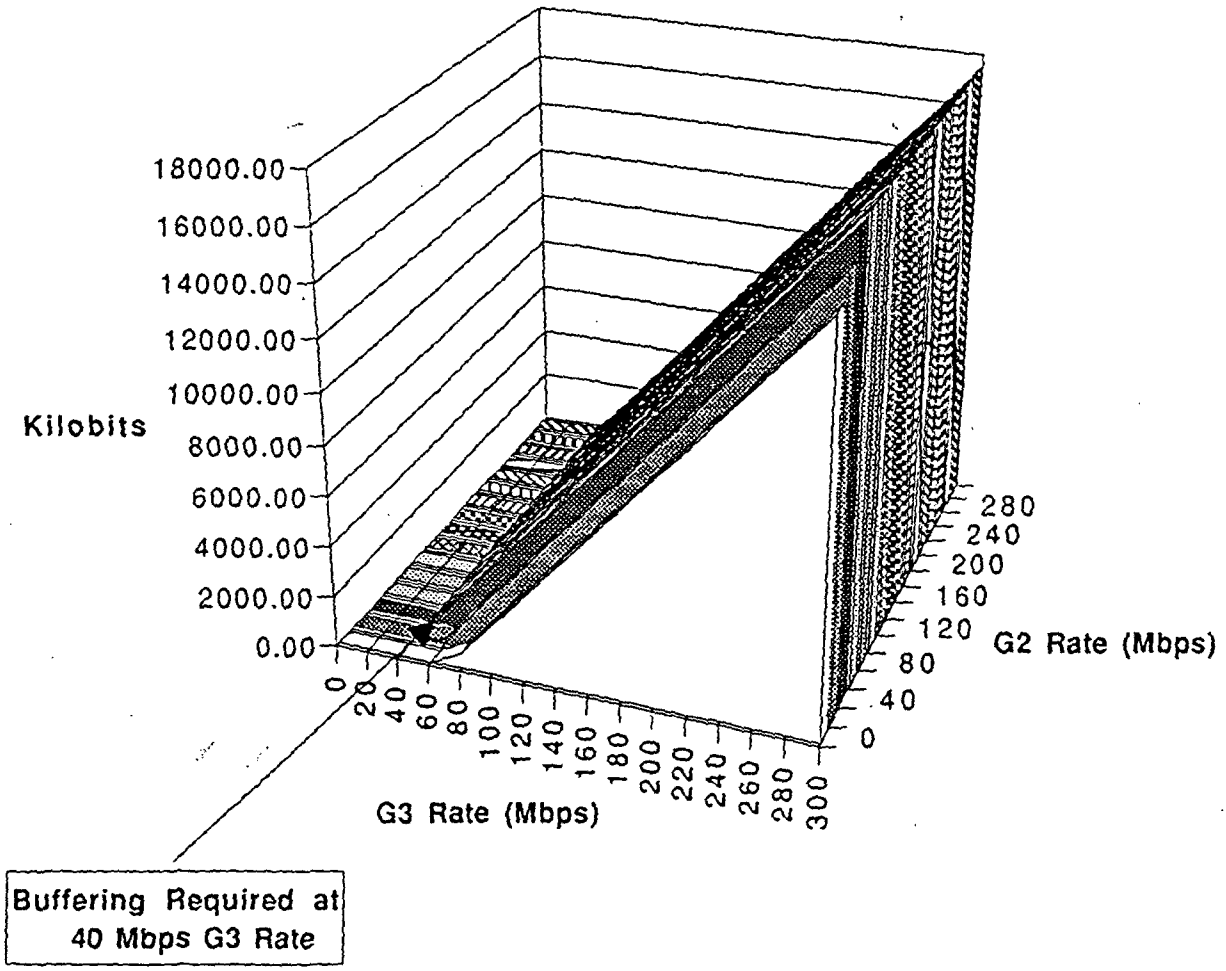
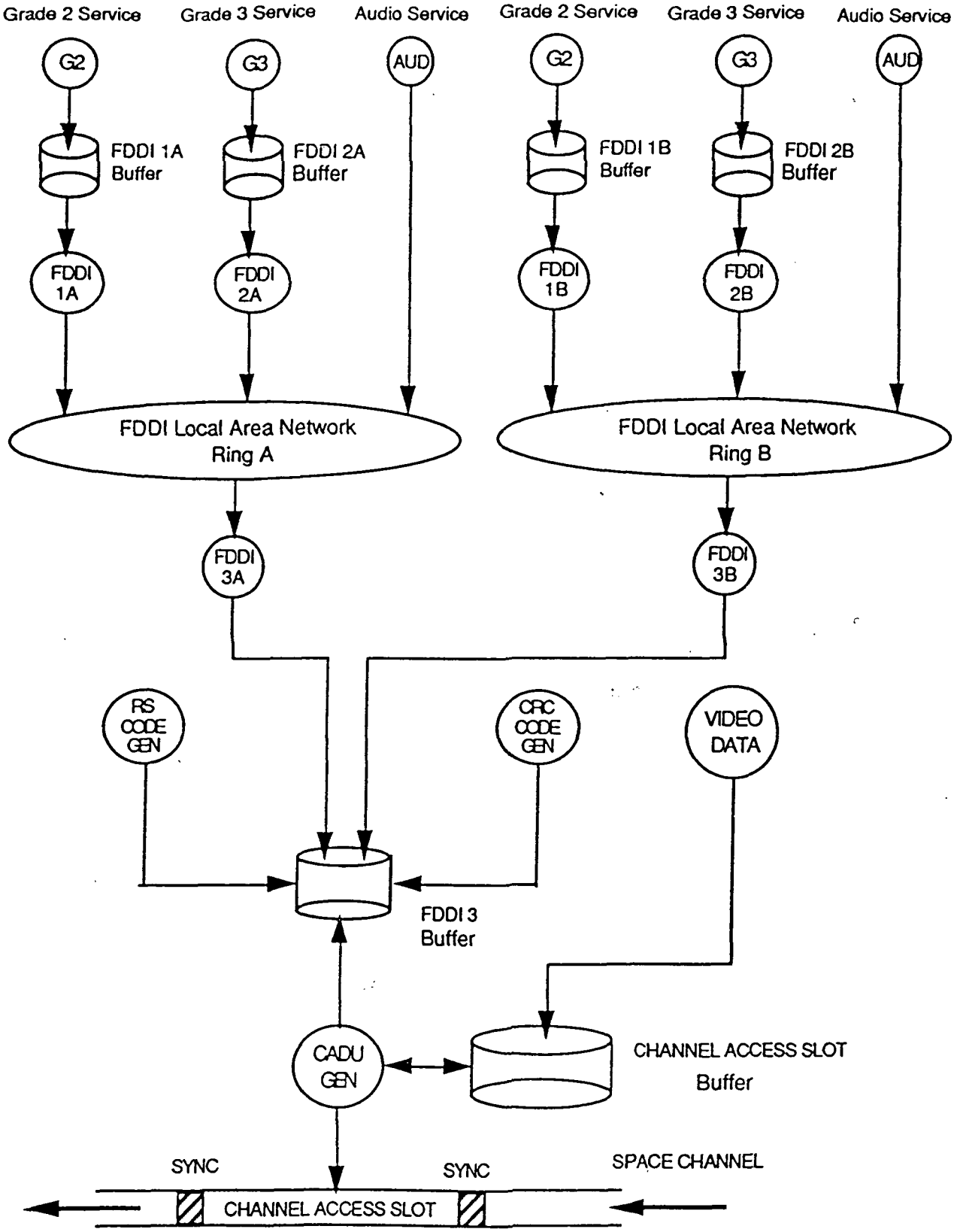


Figure 8.3 Single FDDI Baseline Configuration Grade 3 Buffer Capacity

Figure 5 Dual LAN Simulation Results



Dual FDDI LAN Space Data System

GPSS Model Grade 2 Buffer Capacity

(Dual FDDI, 68 msec)

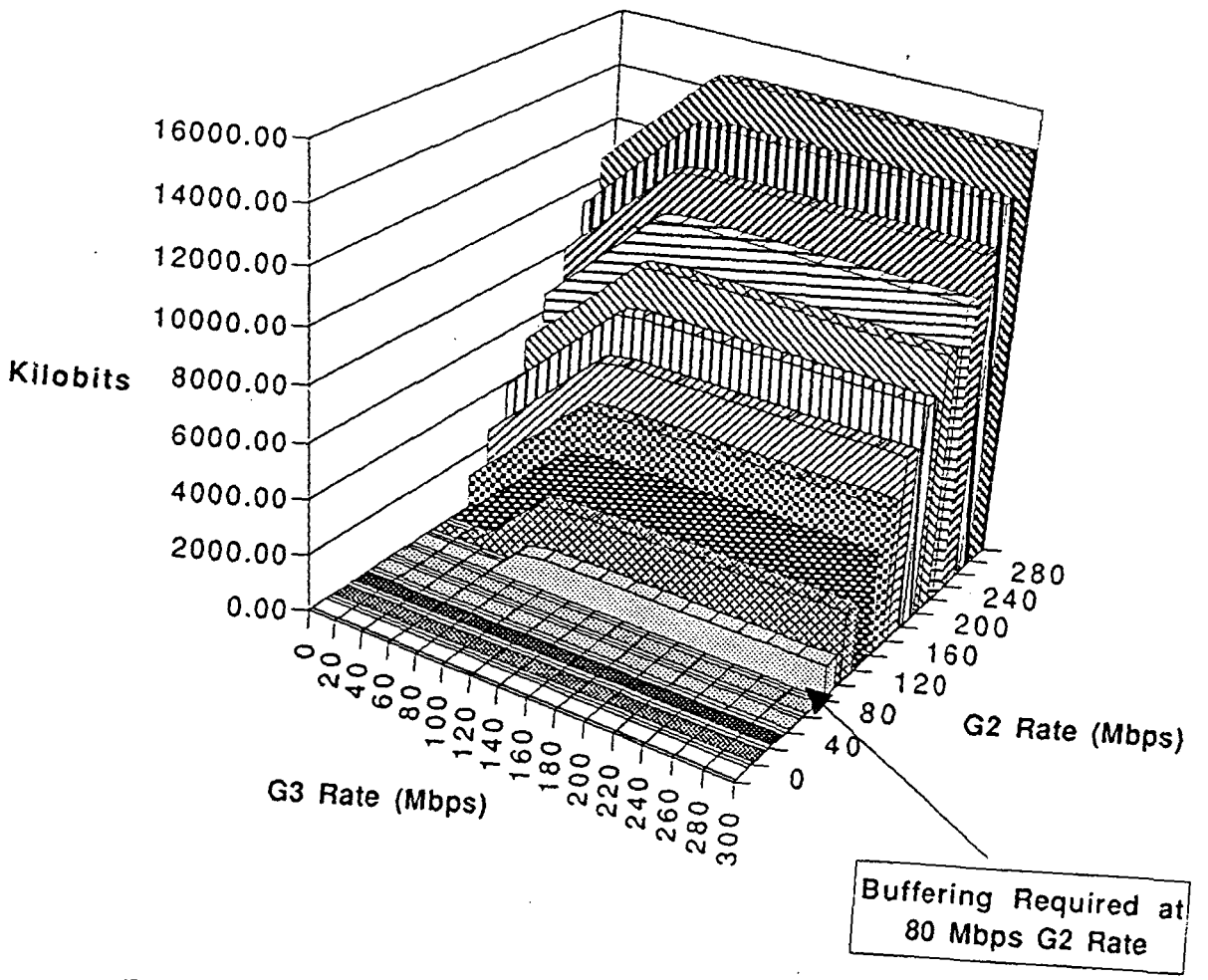


Figure 8.11 Dual FDDI Baseline Configuration Grade 2 Buffer Capacity

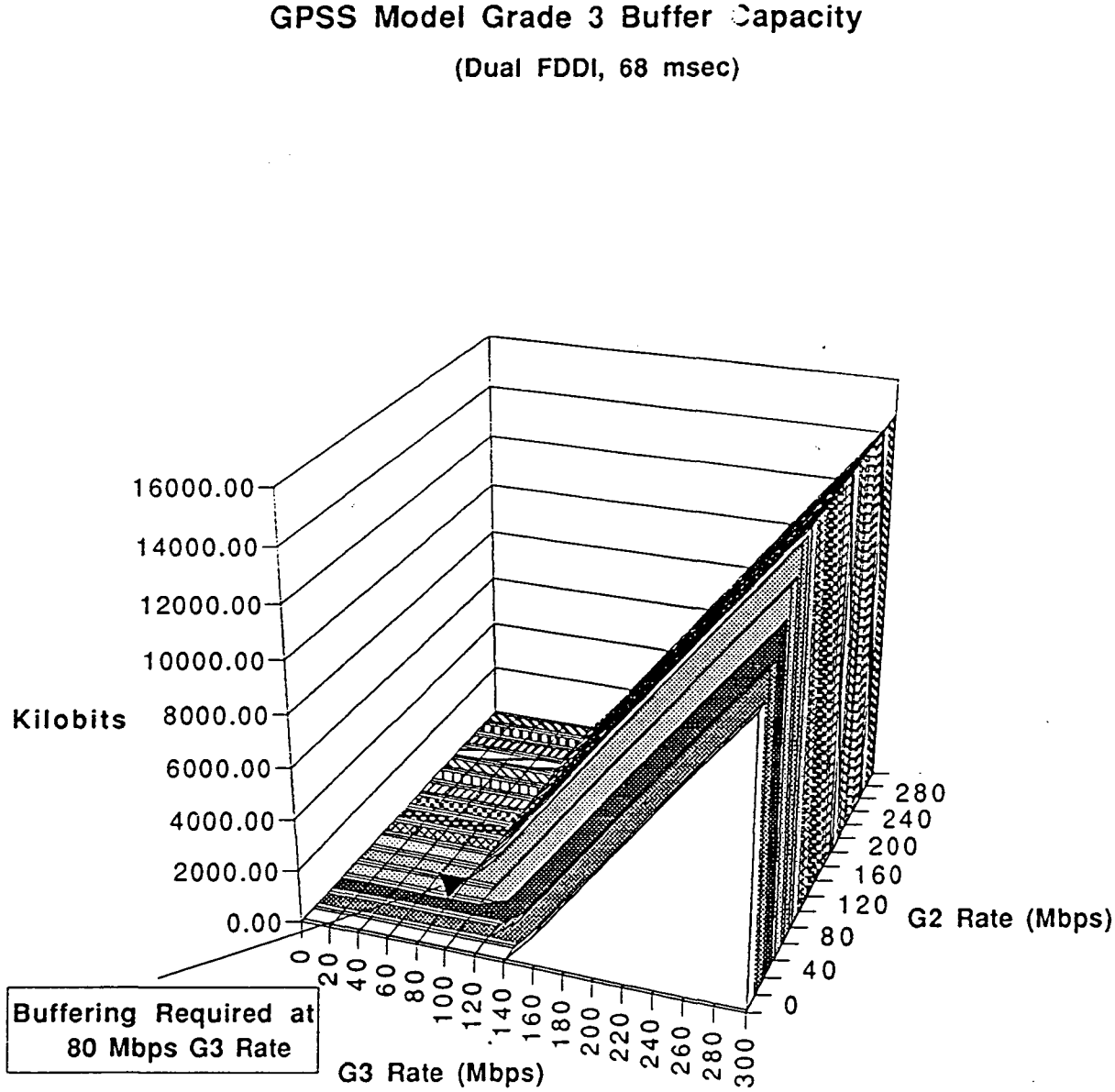


Figure 8.12 Dual FDDI Baseline Configuration Grade 3 Buffer Capacity

GPS68DFM Chart 3

GPSS Model Transfer Frame Fill Data Rate
(Dual FDDI, 68 msec)

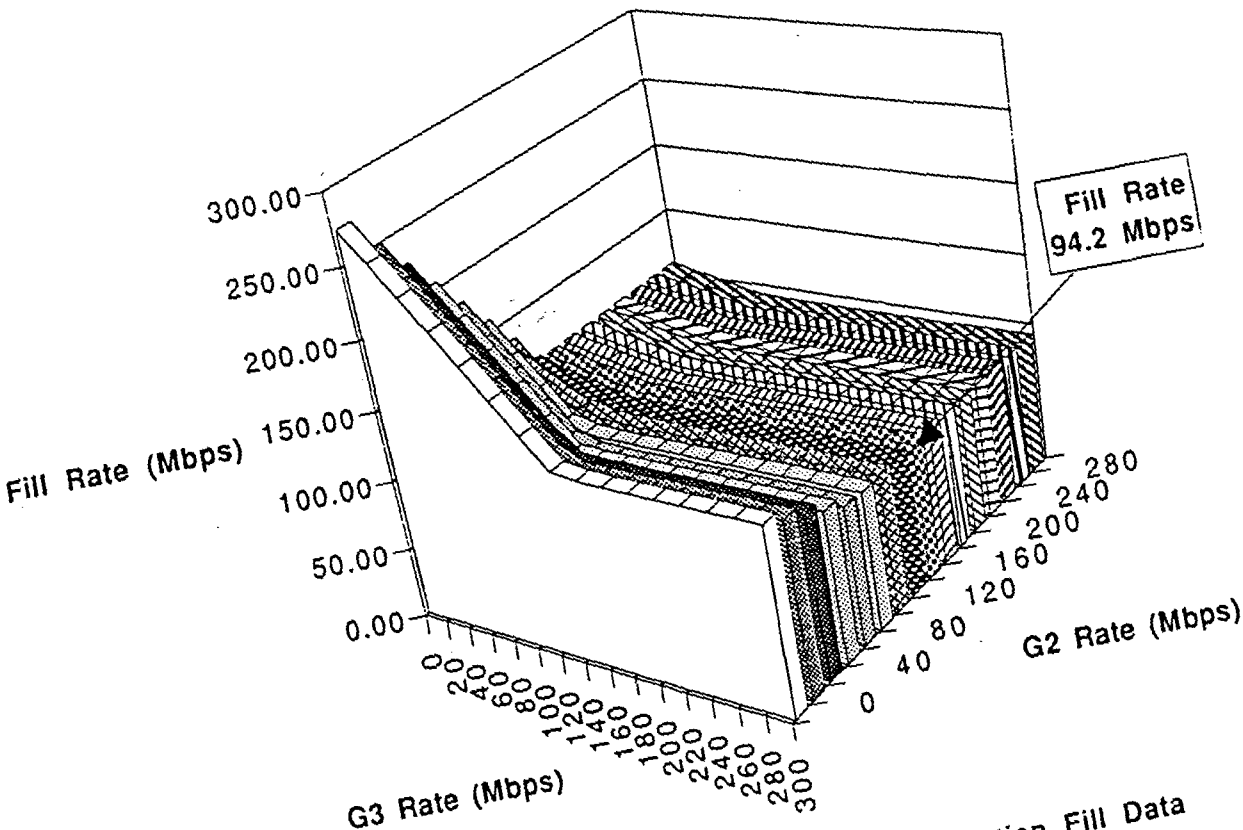
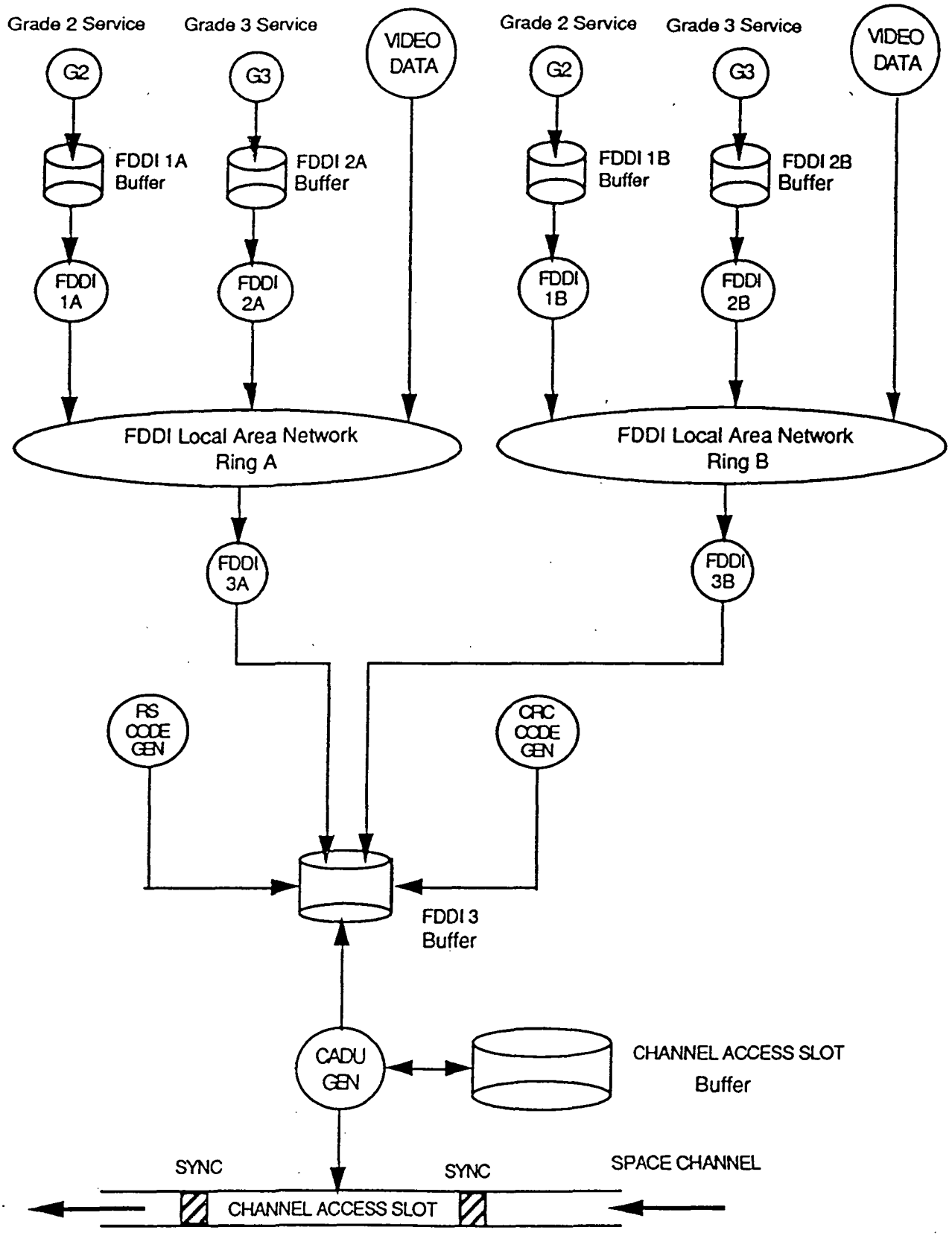


Figure 8.10 Dual FDDI Baseline Configuration Fill Data

Figure 6 Alternate Video Routing



Video Received Through LAN Configuration

GPSS Model Transfer Frame Fill Data Rate
(Dual FDDI with Video)

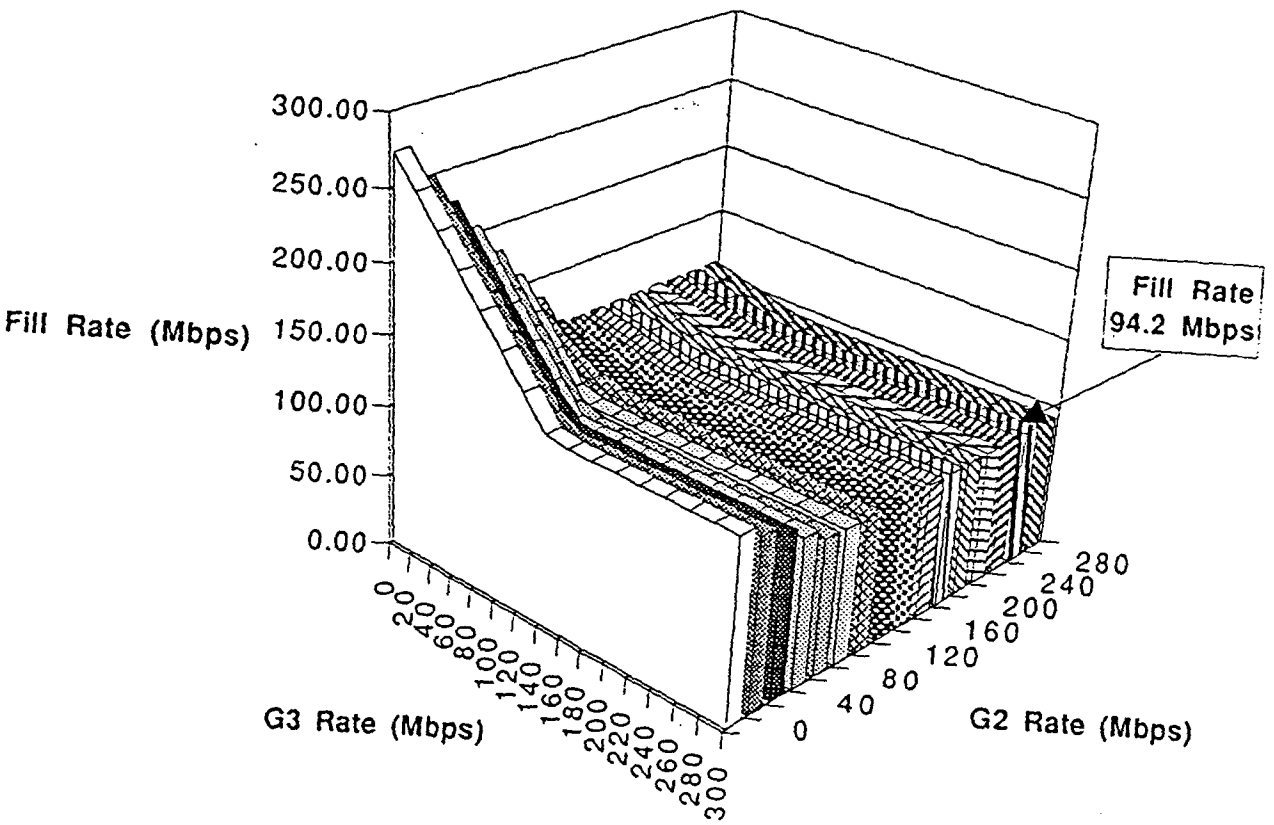


Figure 8.13 Dual FDDI with Video Configuration Transfer Frame Fill Data

GPSS Model Grade 2 Buffer Capacity

(Dual FDDI with Video)

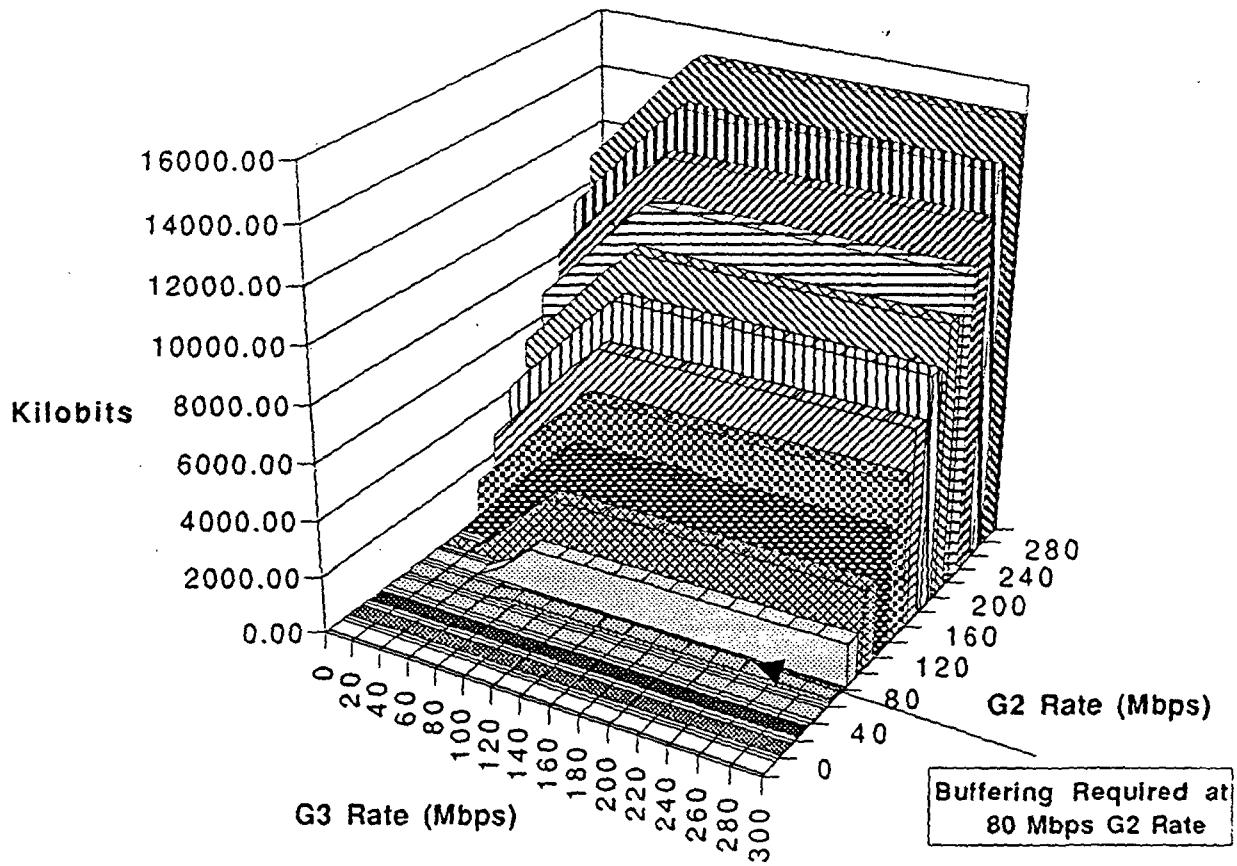


Figure 8.14 Dual FDDI with Video Configuration Grade 2 Buffer Capacity

GPSS Model Grade 3 Buffer Capacity
(Dual FDDI with Video)

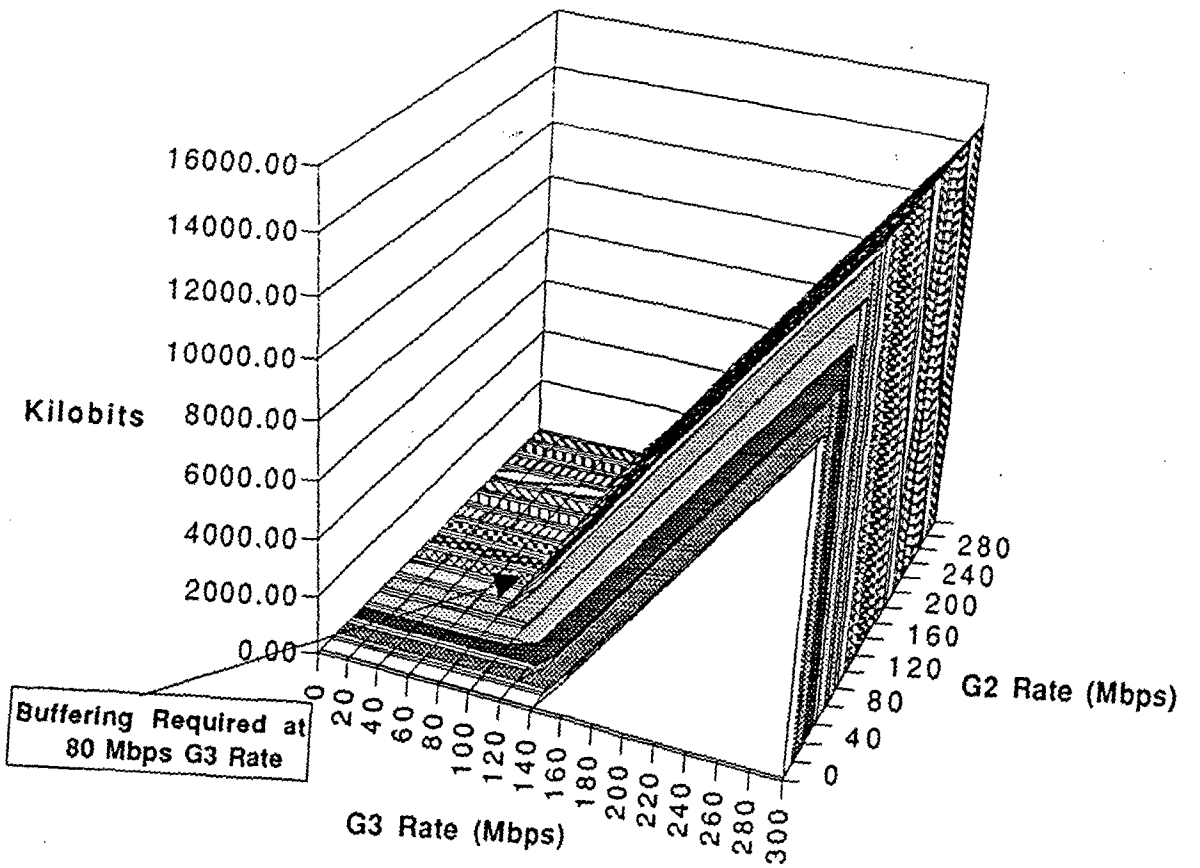


Figure 8.15 Dual FDDI with Video Configuration Grade 3 Buffer Capacity

Sample LPDEMO Listing

```
DUAL300              OBJECTIVE:   MAX      VARIABLES: 40        DATE   02-17-1992
BASIS: NONE          CONSTRAINTS: 38       SLACKS:    4         TIME     15:22:22


RETURN      (VIDbw+..+GS3bw)+2*VIDtf+2*AUDtf+(AUDfta+..+G2Bfta)

VIDEO       VIDbw<=100

AUDIO       AUDbw<=.512

GS2data     GS2bw<=300

GS3data     GS3bw<=300

CADU30ms    (Header+..+FILLtf)=68000

VCDUhdr     Header=320

HDRcrcta    HDRcrc-.0533*Nv=0

VIDscta     -226.7*VIDbw+VIDtf=0

AUDscta     3*AUDtf-AUDfta=0

AUDfddi     -680*AUDbw+AUDfta=0

G2BF1ta     680*GS2bw-GS2bta-G2Afta-G2Bfta=0

G2BF1sz     -.1*GS2bta+G2BUF1=0

G2BF2ta     3*GS2tf+GS2tfb-G2Afta-G2Bfta=0

G2BF2sz     -.1*GS2tfb+G2BUF2=0

G3BF1ta     680*GS3bw-GS3bta-G3Afta-G3Bfta=0

G3BF1sz     -.1*GS3bta+G3BUF1=0

G3BF2ta     3*GS3tf-G3Afta-G3Bfta+GS3tfb=0

G3BF2sz     -.1*GS3tfb+G3BUF2=0

VCDUnum     GS3tf-33.73*Nv=0

G2VCDUnum   GS2tf-29.57*Nc=0

RStb        RStf-4.267*Nc=0

CRCta       CRCtf-.0533*Nv=0

FDDIfma     AUDfta+G3Afta+G2Afta-FDDIAt+HDAfta+TokenA+RingA+FILLA=0

FDDIfmb     G3Bfta+G2Bfta-FDDIBt+HDBfta+TokenB+RingB+FILLB=0
```

```
FDDIrota    -FDDIAt+361*NAf=0
F DIrotb    -FDDIBt+361*NBf=0
FDDIhdra    1.8*NAf-HDAfta=0
F DIhdrb    1.8*NBf-HDBfta=0
FDDItoka    -.88*NAf+TokenA=0
FDDItokb    -.88*NBf+TokenB=0
F DIlata    -10*NAf+RingA=0
FDDIlatb    -10*NBf+RingB=0
F DItota    FDDIAt=68000
FDDItotb    FDDIBt=68000
G2BUFfddi   G2BUF1=0
G BUFfddi   G3BUF1=0
G2BUFtf     G2BUF2=0
G BUFtf     G3BUF2=0
```

DUAL300 SOLUTION IS MAXIMUM RETURN 177088.1288 DATE 02-17-1992
PRIMAL PROBLEM SOLUTION TIME 15:22:37

VARIABLE	STATUS	VALUE	RETURN/UNIT	VALUE/UNIT	NET RETURN
VIDbw	BASIS	100.00000	1.0000000	1.0000000	.00000000
AUDbw	BASIS	.51200000	1.0000000	1.0000000	.00000000
GS2bw	BASIS	.00000000	1.0000000	1.0000000	.00000000
GS3bw	BASIS	192.46307	1.0000000	1.0000000	.00000000
Header	BASIS	320.00000	.00000000	.00000000	.00000000
HDRcrc	BASIS	68.935977	.00000000	.00000000	.00000000
VIDtf	BASIS	22670.000	2.0000000	2.0000000	.00000000
AUDtf	BASIS	116.05333	2.0000000	2.0000000	.00000000
GS2tf	BASIS	.00000000	.00000000	.00000000	.00000000
GS3tf	BASIS	43624.962	.00000000	.00000000	.00000000
RStf	BASIS	.00000000	.00000000	.00000000	.00000000
CRCtf	BASIS	68.935977	.00000000	.00000000	.00000000
FILLtf	BASIS	1131.1123	.00000000	.00000000	.00000000
GS2bta	BASIS	.00000000	.00000000	.00000000	.00000000
G2BUF1	BASIS	.00000000	.00000000	.00000000	.00000000
GS2tfb	NONBASIS	.00000000	.00000000	.00000000	.00000000
G2BUF2	BASIS	.00000000	.00000000	.00000000	.00000000
GS3bta	BASIS	.00000000	.00000000	.00000000	.00000000
G3BUF1	BASIS	.00000000	.00000000	.00000000	.00000000
Nv	BASIS	1293.3579	.00000000	.00000000	.00000000
Nc	NONBASIS	.00000000	.00000000	.00000000	.00000000
AUDfta	BASIS	348.16000	1.0000000	1.0000000	.00000000
G3Afta	BASIS	65263.364	1.0000000	1.0000000	.00000000
G3Bfta	BASIS	65611.524	1.0000000	1.0000000	.00000000
G2Afta	NONBASIS	.00000000	1.0000000	1.0000000	.00000000
G2Bfta	NONBASIS	.00000000	1.0000000	1.0000000	.00000000
FDDIAt	BASIS	68000.000	.00000000	.00000000	.00000000
FDDIBt	BASIS	68000.000	.00000000	.00000000	.00000000
FAf	BASIS	188.36565	.00000000	.00000000	.00000000
FBf	BASIS	188.36565	.00000000	.00000000	.00000000
HDAfta	BASIS	339.05817	.00000000	.00000000	.00000000
HDBfta	BASIS	339.05817	.00000000	.00000000	.00000000
TokenA	BASIS	165.76177	.00000000	.00000000	.00000000
TokenB	BASIS	165.76177	.00000000	.00000000	.00000000
RingA	BASIS	1883.6565	.00000000	.00000000	.00000000
RingB	BASIS	1883.6565	.00000000	.00000000	.00000000
GS3tfb	NONBASIS	.00000000	.00000000	.00000000	.00000000
G3BUF2	BASIS	.00000000	.00000000	.00000000	.00000000
FILLA	NONBASIS	.00000000	.00000000	1.0014706	-1.0014706
FILLB	NONBASIS	.00000000	.00000000	1.0014706	-1.0014706
S.1	NONBASIS	.00000000	.00000000	454.40000	-454.40000
S.2	NONBASIS	.00000000	.00000000	453.33333	-453.33333
S.3	BASIS	300.00000	.00000000	.00000000	.00000000
S.4	BASIS	107.53693	.00000000	.00000000	.00000000

DUAL300 SOLUTION IS MAXIMUM RETURN 177088.1288 DATE 02-17-1992
RIGHT-HAND-SIDE RANGES TIME 15:23:03

CONSTRAINT	STATUS	DUAL VALUE	RHS VALUE	MINIMUM	MAXIMUM
IDEO	BINDING	454.40000	100.00000	.00000000	104.98947
UDIO	BINDING	453.33333	.51200000	.00000000	96.487535
GS2data	NONBINDING	.00000000	300.00000	.00000000	NONE
GS3data	NONBINDING	.00000000	300.00000	192.46307	NONE
ADU30ms	NONBINDING	.00000000	68000.000	66868.888	NONE
vCDUhdr	NONBINDING	.00000000	320.00000	.00000000	1451.1123
HDRcrcta	NONBINDING	.00000000	.00000000	-68.935977	1131.1123
IDscta	BINDING	2.0000000	.00000000	-22670.000	1131.1123
UDscta	BINDING	.66666667	.00000000	-348.16000	3393.3370
AUDfddi	BINDING	.66519608	.00000000	-348.16000	65263.364
2BF1ta	BINDING	.00147059	.00000000	.00000000	204000.00
2BF1sz	BINDING	-.01470588	.00000000	-20400.000	.00000000
G2BF2ta	NONBINDING	.00000000	.00000000	.00000000	.00000000
G2BF2sz	NONBINDING	.00000000	.00000000	.00000000	.00000000
3BF1ta	BINDING	.00147059	.00000000	-130874.89	73125.113
G3BF1sz	BINDING	-.01470588	.00000000	-7312.5113	.00000000
G3BF2ta	NONBINDING	.00000000	.00000000	-130874.89	3382.6466
3BF2sz	NONBINDING	.00000000	.00000000	.00000000	.00000000
CDUnum	NONBINDING	.00000000	.00000000	-357902.62	43624.962
CVCDUnum	NONBINDING	.00000000	.00000000	.00000000	NONE
Sta	NONBINDING	.00000000	.00000000	.00000000	1131.1123
RCta	NONBINDING	.00000000	.00000000	-68.935977	1131.1123
FDDIfma	BINDING	1.0014706	.00000000	-65263.364	3382.6466
FDDIfmb	BINDING	1.0014706	.00000000	-65611.524	3382.6466
DDIrota	BINDING	-.03517631	.00000000	-68000.000	1858050.0
DDIrotb	BINDING	-.03517631	.00000000	-68000.000	1867962.1
FDDIhdra	BINDING	1.0014706	.00000000	-65263.364	339.05817
DDIhdrb	BINDING	1.0014706	.00000000	-65611.524	339.05817
DDItoka	BINDING	-1.0014706	.00000000	-165.76177	65263.364
FDDItokb	BINDING	-1.0014706	.00000000	-165.76177	65611.524
DDIlata	BINDING	-1.0014706	.00000000	-1883.6565	65263.364
DDIlatb	BINDING	-1.0014706	.00000000	-1883.6565	65611.524
FDDItota	BINDING	.96629428	68000.000	360.83418	71505.786
FDDItotb	BINDING	.96629428	68000.000	.00000000	71505.786
2BUFfddi	BINDING	.01470588	.00000000	.00000000	20400.000
G3BUFfddi	BINDING	.01470588	.00000000	.00000000	7312.5113
G2BUFtf	NONBINDING	.00000000	.00000000	.00000000	NONE
G3BUFtf	NONBINDING	.00000000	.00000000	.00000000	NONE

Sample GPSS Listing

```
      SIMULATE
*
*
*     THE PACKET LENGTH DURATION IN MICROSECONDS SETS THE
*     TRANSFER RATE FOR THE FDDI RING
*
PKTLEN  EQU     10          1000 BIT PACKET (USEC)   (100 MBPS)
        REAL    &TPTA       TOKEN PASSING TIME (USEC)   RING A
        REAL    &TRTA       TOKEN ROTATION TIME (USEC)
        REAL    &TTRTA      TARGET TOKEN ROTATION TIME (USEC)
        REAL    &TPTB       TOKEN PASSING TIME (USEC)   RING B
        REAL    &TRTB       TOKEN ROTATION TIME (USEC)
        REAL    &TTRTB      TARGET TOKEN ROTATION TIME (USEC)
        REAL    &TTRT1A     STATION 1 ASYNC TTRT (USEC) RING A
        REAL    &TTRT2A       "    2  "    "   "
        REAL    &TTRT5A       "    5  "    "   "
        REAL    &TTRT1B     STATION 1 ASYNC TTRT (USEC) RING B
        REAL    &TTRT2B       "    2  "    "   "
        REAL    &TTRT3B       "    3  "    "   "
        REAL    &THT1A      STATION 1 TOKEN HOLD TIME (USEC)
        REAL    &THT2A        "    2  "    "   "    "
        REAL    &THT3A        "    3  "    "   "    "
        REAL    &THT1B      STATION 1 TOKEN HOLD TIME (USEC)
        REAL    &THT2B        "    2  "    "   "    "
        REAL    &THT3B        "    3  "    "   "    "
        REAL    &FILL       FILL DATA PACKETS
        REAL    &FP         CHANNEL ACCESS DATA UNIT FDDI DATA
        REAL    &FPA        CHANNEL ACCESS DATA UNIT FDDI DATA
        REAL    &FPB        CHANNEL ACCESS DATA UNIT FDDI DATA
        REAL    &G2D        GRADE 2 DATA RECEIVED AT CADU GEN
        REAL    &G2A        GRADE 2 DATA RECEIVED THRU RING A
        REAL    &G2B        GRADE 2 DATA RECEIVED THRU RING B
        REAL    &G3D        GRADE 3 DATA RECEIVED AT CADU GEN
        REAL    &G3A        GRADE 3 DATA RECEIVED THRU RING A
        REAL    &G3B        GRADE 3 DATA RECEIVED THRU RING B
        REAL    &VP         CHANNEL ACCESS DATA UNIT VIDEO DATA
        REAL    &DZ         CHANNEL ACCESS DATA UNIT ZONE
        REAL    &DZT          "     "     "   "  "  TOT.
        REAL    &RS         REED SOLOMON CHECK CODE FOR G2 DATA
        REAL    &RST          "     "     "   "  "  "  " TOT.
        REAL    &VH         VCDU AND CVCDU HEADER BITS PER CADU
        REAL    &VHT          "   "   "   "   "   "  " TOT.
        REAL    &CRC        VCDU HEADER & DATA CRC CODE BITS (G5)
        REAL    &CRCT         "   "     "   "   "   "  TOT.
        INTEGER &PS1A       PATH SERVICE BUFFER COUNT   RING A
        INTEGER &PS2A         "    "       "    "
        INTEGER &PS3A         "    "       "    "
        INTEGER &PS1B       PATH SERVICE BUFFER COUNT   RING B
        INTEGER &PS2B         "    "       "    "
        INTEGER &PS3B         "    "       "    "
        INTEGER &TRF1A      TOKEN RECEIVED FLAG STATION 1
        INTEGER &TRF2A        "     "       "    "    2
        INTEGER &TRF3A        "     "       "    "    3
        INTEGER &TRF1B      TOKEN RECEIVED FLAG STATION 1
        INTEGER &TRF2B        "     "       "    "    2
```



```
        INTEGER &TRF3B        "     "       "    "    3
        INTEGER &FP1A       FDDI PACKET TRANSFER COUNT
        INTEGER &FP1B       FDDI PACKET TRANSFER COUNT
        INTEGER &FP1A       FDDI STATION 1 PACKET TRANSFER COUNT
        INTEGER &FP2A       FDDI STATION 2 PACKET TRANSFER COUNT
        INTEGER &FP3A       FDDI STATION 3 PACKET TRANSFER COUNT
        INTEGER &FP1B       FDDI STATION 1 PACKET TRANSFER COUNT
        INTEGER &FP2B       FDDI STATION 2 PACKET TRANSFER COUNT
        INTEGER &FP3B       FDDI STATION 3 PACKET TRANSFER COUNT
        INTEGER &VPT        VIDEO PACKET TRANSFER COUNT
        INTEGER &GS2        INCREMENTS GRADE 2 SERVICE DATA RATE
        INTEGER &GS3        INCREMENTS GRADE 3 SERVICE DATA RATE
*
*           FDDI RING STATION 1 RECEIVES PATH SERVICE 1 DATA
*
FDS1A   TEST G  &PS1A,0,RELTOK1A    DOES STATION 1 REQUEST TOKEN?
        BLET    &THT1A=0            INITIALIZE TOKEN HOLD TIME
LP1A    TEST G  &TTRTA,&TRTA,EXPIRED  HAS TTRT EXPIRED?
        TEST G  &TTRT1A,&THT1A,RELTOK1A  TOKEN HOLD TIME EXPIRED?
T1A     ADVANCE PKTLEN              PACKET TRANSMISSION TIME
        BLET    &FP1A=&FP1A+1       FDDI PACKET TRANSFER COUNT
        BLET    &THT1A=&THT1A+PKTLEN  UPDATE TOKEN HOLD TIME
        BLET    &TRTA=&TRTA+PKTLEN  UPDATE TOKEN ROTATION TIME
        BLET    &PS1A=&PS1A-1       DECREMENT THE ASYNC DATA BUFFER
        TEST G  &PS1A,0,RELTOK1A    HAS ALL DATA BEEN SENT?
        TRANSFER ,LP1A
RELTOK1A BLET   &TRF1A=1            SET TOKEN RECEIVE FLAG
        RELEASE TOKENA              RELEASE TOKEN
        TERMINATE
*
*           FDDI RING STATION 2 FOR PATH SERVICE 2
*
FDS2A   TEST G  &PS2A,0,RELTOK2A    DOES STATION 2 REQUEST TOKEN?
        BLET    &THT2A=0            INITIALIZE TOKEN HOLD TIME
LP2A    TEST G  &TTRTA,&TRTA,EXPIRED  HAS TTRT EXPIRED?
        TEST G  &TTRT2A,&THT2A,RELTOK2A  TOKEN HOLD TIME EXPIRED?
T2A     ADVANCE PKTLEN              PACKET TRANSMISSION TIME
        BLET    &FP2A=&FP2A+1       FDDI PACKET TRANSFER COUNT
        BLET    &THT2A=&THT2A+PKTLEN  UPDATE TOKEN HOLD TIME
        BLET    &TRTA=&TRTA+PKTLEN  UPDATE TOKEN ROTATION TIME
        BLET    &PS2A=&PS2A-1       DECREMENT ASYNC DATA BUFFER
        TEST G  &PS2A,0,RELTOK2A    HAS ALL DATA BEEN SENT?
        TRANSFER ,LP2A
RELTOK2A BLET   &TRF2A=1            SET TOKEN RECEIVE FLAG
        RELEASE TOKENA              RELEASE TOKEN
        TERMINATE
*
*           FDDI RING STATION 3 FOR PATH SERVICE 3
*
FDS3A   TEST G  &PS3A,0,RELTOK3A    DOES STATION 3 REQUEST TOKEN?
        BLET    &THT3A=0            INITIALIZE TOKEN HOLD TIME
LP3A    TEST G  &TTRTA,&TRTA,EXPIRED  HAS TTRT EXPIRED?
        TEST G  &TTRT3A,&THT3A,RELTOK3A  TOKEN HOLD TIME EXPIRED?
T3A     ADVANCE PKTLEN              PACKET TRANSMISSION TIME
        BLET    &FP3A=&FP3A+1       FDDI PACKET TRANSFER COUNT
```

```
         BLET    &THT5A=&TH15A+PKTLEN  UPDATE TOKEN HOLD TIME
         BLET    &TRTA=&TRTA+PKTLEN    UPDATE TOKEN ROTATION TIME
         BLET    &PS5A=&PS5A-1         DECREMENT ASYNC DATA BUFFER
         TEST G  &PS5A,0,RELTOK5A      HAS ALL DATA BEEN SENT?
         TRANSFER ,LP5A
RELTOK5A BLET    &TRF5A=1              SET TOKEN RECEIVE FLAG
         RELEASE TOKENA                RELEASE TOKEN
         TERMINATE
*
         GENERATE                      GENERATE THE TOKEN
         SEIZE   TOKENA
LOOPA    TEST G  &TRF1A,0,FDS1A        HAS STATION 1 RECEIVED TOKEN?
         ADVANCE &TPTA                 TOKEN PASSING TIME
         BLET    &TRTA=&TRTA+&TPTA     UPDATE TOKEN ROTATION TIME
         TEST G  &TRF2A,0,FDS2A        HAS STATION 2 RECEIVED TOKEN?
         ADVANCE &TPTA                 TOKEN PASSING TIME
         BLET    &TRTA=&TRTA+&TPTA     UPDATE TOKEN ROTATION TIME
         TEST G  &TRF5A,0,FDS5A        HAS STATION 5 RECEIVED TOKEN?
         ADVANCE &TPTA                 TOKEN PASSING TIME
         BLET    &TRTA=&TRTA+&TPTA     UPDATE TOKEN ROTATION TIME
EXPIRED  BLET    &TRTA=0               RESET EXPIRED TOKEN
         BLET    &TRF1A=0              IF ALL STATIONS HAVE RECEIVED
         BLET    &TRF2A=0              TOKEN, START OVER
         BLET    &TRF5A=0
         TRANSFER ,LOOPA
*
*-------------------------FDDI RING B ------------------------------
*
*           FDDI RING STATION 1 RECEIVES PATH SERVICE 1 DATA
*
FDS1B    TEST G  &PS1B,0,RELTOK1B      DOES STATION 1 REQUEST TOKEN?
         BLET    &TH1B=0               INITIALIZE TOKEN HOLD TIME
LP1B     TEST G  &TTRTB,&TRTB,EXPIRE   HAS TTRT EXPIRED?
         TEST G  &TTRT1B,&TH1B,RELTOK1B  TOKEN HOLD TIME EXPIRED?
T1B      ADVANCE PKTLEN                PACKET TRANSMISSION TIME
         BLET    &FP1B=&FP1B+1         FDDI PACKET TRANSFER COUNT
         BLET    &TH1B=&TH1B+PKTLEN    UPDATE TOKEN HOLD TIME
         BLET    &TRTB=&TRTB+PKTLEN    UPDATE TOKEN ROTATION TIME
         BLET    &PS1B=&PS1B-1         DECREMENT THE ASYNC DATA BUFFER
         TEST G  &PS1B,0,RELTOK1B      HAS ALL DATA BEEN SENT?
         TRANSFER ,LP1B
RELTOK1B BLET    &TRF1B=1              SET TOKEN RECEIVE FLAG
         RELEASE TOKENB                RELEASE TOKEN
         TERMINATE
*
*           FDDI RING STATION 2 FOR PATH SERVICE 2
*
FDS2B    TEST G  &PS2B,0,RELTOK2B      DOES STATION 2 REQUEST TOKEN?
         BLET    &TH12B=0              INITIALIZE TOKEN HOLD TIME
LP2B     TEST G  &TTRTB,&TRTB,EXPIRE   HAS TTRT EXPIRED?
         TEST G  &TTRT2B,&TH12B,RELTOK2B  TOKEN HOLD TIME EXPIRED?
T2B      ADVANCE PKTLEN                PACKET TRANSMISSION TIME
         BLET    &FP2B=&FP2B+1         FDDI PACKET TRANSFER COUNT
         BLET    &TH12B=&TH12B+PKTLEN  UPDATE TOKEN HOLD TIME
         BLET    &TRTB=&TRTB+PKTLEN    UPDATE TOKEN ROTATION TIME
```



```
         BLET    &PS2B=&PS2B-1         DECREMENT ASYNC DATA BUFFER
         TEST G  &PS2B,0,RELTOK2B      HAS ALL DATA BEEN SENT?
         TRANSFER ,LP2B
RELTOK2B BLET    &TRF2B=1              SET TOKEN RECEIVE FLAG
         RELEASE TOKENB                RELEASE TOKEN
         TERMINATE
*
*           FDDI RING STATION 5 FOR PATH SERVICE 5
*
FDS5B    TEST G  &PS5B,0,RELTOK5B      DOES STATION 5 REQUEST TOKEN?
         BLET    &TH15B=0              INITIALIZE TOKEN HOLD TIME
LP5B     TEST G  &TTRTB,&TRTB,EXPIRE   HAS TTRT EXPIRED?
         TEST G  &TTRT5B,&TH15B,RELTOK5B  TOKEN HOLD TIME EXPIRED?
T5B      ADVANCE PKTLEN                PACKET TRANSMISSION TIME
         BLET    &FP5B=&FP5B+1         FDDI PACKET TRANSFER COUNT
         BLET    &TH15B=&TH15B+PKTLEN  UPDATE TOKEN HOLD TIME
         BLET    &TRTB=&TRTB+PKTLEN    UPDATE TOKEN ROTATION TIME
         BLET    &PS5B=&PS5B-1         DECREMENT ASYNC DATA BUFFER
         TEST G  &PS5B,0,RELTOK5B      HAS ALL DATA BEEN SENT?
         TRANSFER ,LP5B
RELTOK5B BLET    &TRF5B=1              SET TOKEN RECEIVE FLAG
         RELEASE TOKENB                RELEASE TOKEN
         TERMINATE
*
         GENERATE                      GENERATE THE TOKEN
         SEIZE   TOKENB
LOOPB    TEST G  &TRF1B,0,FDS1B        HAS STATION 1 RECEIVED TOKEN?
         ADVANCE &TPTB                 TOKEN PASSING TIME
         BLET    &TRTB=&TRTB+&TPTB     UPDATE TOKEN ROTATION TIME
         TEST G  &TRF2B,0,FDS2B        HAS STATION 2 RECEIVED TOKEN?
         ADVANCE &TPTB                 TOKEN PASSING TIME
         BLET    &TRTB=&TRTB+&TPTB     UPDATE TOKEN ROTATION TIME
         TEST G  &TRF5B,0,FDS5B        HAS STATION 5 RECEIVED TOKEN?
         ADVANCE &TPTB                 TOKEN PASSING TIME
         BLET    &TRTB=&TRTB+&TPTB     UPDATE TOKEN ROTATION TIME
EXPIRE   BLET    &TRTB=0               RESET EXPIRED TOKEN
         BLET    &TRF1B=0              IF ALL STATIONS HAVE RECEIVED
         BLET    &TRF2B=0              TOKEN, START OVER
         BLET    &TRF5B=0
         TRANSFER ,LOOPB
*
*        GENERATE  40                   VIDEO DATA GENERATOR (25  KBPS)
*        BLET    &VPT=&VPT+1           SEND VIDEO DATA PACKET
*        TERMINATE
*
         GENERATE  34                   SEND CHANNEL ACCESS DATA UNIT
         BLET    &G2A=&FP1A-&G2A       NUMBER OF G2 PACKETS RECEIVED
         BLET    &G5A=&FP2A-&G5A       NUMBER OF G5 PACKETS RECEIVED
         BLET    &G2B=&FP1B-&G2B       NUMBER OF G2 PACKETS RECEIVED
         BLET    &G5B=&FP2B-&G5B       NUMBER OF G5 PACKETS RECEIVED
         BLET    &G2D=&G2A+&G2B        GRADE 2 DATA PACKETS
         BLET    &RS=&G2D*0.144274     RS CODE BITS PER G2 PACKET
         BLET    &G5D=&G5A+&G5B        GRADE 5 DATA PACKETS
         BLET    &CRC=&G5D*0.007942B   CRC CODE BITS PER G5 PACKET
         BLET    &FPTA=&FP1A+&FP2A+&FP5A  COMPUTE NUMBER OF FDDI
```

```
         BLET    &FPA=&FPTA-&FPA         PACKETS SENT OVER RING A
         BLET    &FPTB=&FP1B+&FP2B+&FPSB  COMPUTE NUMBER OF FDDI
         BLET    &FPB=&FPTB-&FPB         PACKETS SENT OVER RING B
         BLET    &FP=&FPA+&FPB           # OF FDDI PACKETS IN CADU
         BLET    &VP=&VPT-&VP            COMPUTE # VIDEO PACKETS RCVD
         BLET    &DZ=&FP*&VP             COMPUTE CADU DATA ZONE
         BLET    &FILL=&FILL+10.2-&DZ-&RS-&VH-&CRC   FILL DATA
         BLET    &VHT=&VHT+&VH           ACCUMULATE VCDU HEADER BITS
         BLET    &RST=&RST+&RS           ACCUMULATE RS CHECK BITS
         BLET    &CRCT=&CRCT+&CRC        ACCUMULATE CRC CODE BITS
         BLET    &DZT=&DZT+&DZ           ACCUMULATE CADU VALID DATA
         BLET    &G2A=&FP1A              SAVE LAST RING A PKT COUNT
         BLET    &G3A=&FP2A
         BLET    &FPA=&FP1A+&FP2A+&FPSA
         BLET    &G2B=&FP1B              SAVE LAST RING B PKT COUNT
         BLET    &G3B=&FP2B
         BLET    &FPB=&FP1B+&FP2B+&FPSB
         BLET    &VP=&VPT                SAVE LAST VIDEO PACKET COUNT
         TERMINATE
*
         GENERATE  60000                 SIMULATE FOR 60 MILLISECONDS
         TERMINATE 1
*
         DO      &GS2=8,15               GRADE 2 SERVICE DATA RATE
         DO      &GS3=8,15               GRADE 3 SERVICE DATA RATE
*
         LET     &PS1A=680*&GS2          PATH SERVICE 1 OFFERED LOAD
         LET     &PS2A=680*&GS3           "    "   2   "      "
         LET     &PS3A=34                 "    "   3   "      "
         LET     &PS1B=680*&GS2          PATH SERVICE 1 OFFERED LOAD
         LET     &PS2B=680*&GS3           "    "   2   "      "
         LET     &PS3B=34                 "    "   3   "      "
         LET     &TIRTA=400              TARGET TOKEN ROTATION TIME
         LET     &TIRT1A=75              STATION 1 ASYNC TIRT (USEC)
         LET     &TIRT2A=75                "   2   "   "   "
         LET     &TIRT3A=200               "   3   "   "   "
         LET     &TPTA=0.1               TOKEN PASSING TIME (USEC)
         LET     &TIRTB=400              TARGET TOKEN ROTATION TIME
         LET     &TIRT1B=75              STATION 1 ASYNC TIRT (USEC)
         LET     &TIRT2B=75                "   2   "   "   "
         LET     &TIRT3B=200               "   3   "   "   "
         LET     &TPTB=0.1               TOKEN PASSING TIME (USEC)
         LET     &VH=0.048               VCDU AND CVCDU HEADER (KBITS)

         LET     &FILL=0.0               INITIALIZE FILL DATA TO ZERO
         LET     &FP=0.0                   "    CADU FDDI DATA
         LET     &FPA=0.0                  "    FDDI DATA RING A
         LET     &FPB=0.0                  "    FDDI DATA RING B
         LET     &G2D=0.0                  "    G2 DATA IN CADU
         LET     &G2A=0.0                  "    G2 DATA THRU RING A
         LET     &G2B=0.0                  "    G2 DATA THRU RING B
         LET     &G3D=0.0                  "    G3 DATA IN CADU
         LET     &G3A=0.0                  "    G3 DATA THRU RING A
         LET     &G3B=0.0                  "    G3 DATA THRU RING B
         LET     &VP=0.0                   "    CADU VIDEO DATA
```



```
         LET     &DZ=0.0                   "    CADU DATA ZONE
         LET     &DZT=0.0                  "    CADU DATA ZONE
         LET     &RST=0.0                  "    RS CODE     "
         LET     &RS=0.9                   "    RS CODE     "
         LET     &VHT=0.0                  "    HEADER      "
         LET     &CRC=0.0                  "    CRC CODE    "
         LET     &CRCT=0.0                 "    CRC CODE    "
         LET     &VPT=0                  INITIALIZE VIDEO PKT COUNTER
         LET     &FP1A=0                 INITIALIZE FDDI PKT CTR 1A
         LET     &FP2A=0                   "    "   "   "  2A
         LET     &FPSA=0                   "    "   "   "  3A
         LET     &FP1B=0                   "    "   "   "  1B
         LET     &FP2B=0                   "    "   "   "  2B
         LET     &FPSB=0                   "    "   "   "  3B
*
         START   1,NP
         PUTPIC  FILE=OUT,LINES=3,(&GS2*20,&GS3*20)

------------G2 RATE = ####  MBPS      G3 RATE = ####  MBPS -------------

         PUTPIC  FILE=OUT,LINES=2,(&PS1A,N(T1A)=0.014705882)
           G2    PATH SERVICE 1A MAX BUFFER SIZE = ####   KILOBITS
                       THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=2,(&PS2A,N(T2A)=0.014705882)
           G3    PATH SERVICE 2A MAX BUFFER SIZE = ####   KILOBITS
                       THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=2,(&PS3A,N(T3A)=0.014705882)
         VID/AUD PATH SERVICE 3A MAX BUFFER SIZE = ####   KILOBITS
                       THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&FPTA=0.014705882)
                       FDDI A THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=2,(&PS1B,N(T1B)=0.014705882)
           G2    PATH SERVICE 1B MAX BUFFER SIZE = ####   KILOBITS
                       THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=2,(&PS2B,N(T2B)=0.014705882)
           G3    PATH SERVICE 2B MAX BUFFER SIZE = ####   KILOBITS
                       THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=2,(&PS3B,N(T3B)=0.014705882)
         VID/AUD PATH SERVICE 3B MAX BUFFER SIZE = ####   KILOBITS
                       THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&FPTB=0.014705882)
                       FDDI B THROUGHPUT   =  ####.###  MBPS
*        PUTPIC  FILE=OUT,LINES=1,(&VPT=0.014705882)
*            VIDEO DATA  THROUGHPUT   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&DZT=0.014705882)
                       CADU VALID DATA    =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&VHT=0.014705882)
                       CADU HEADER DATA   =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&RST=0.014705882)
                       CADU RS CHECK CODE =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&CRCT=0.014705882)
                       CADU CRC CODE DATA =  ####.###  MBPS
         PUTPIC  FILE=OUT,LINES=1,(&FILL=0.014705882)
                       CADU  FILL DATA    =  ####.###  MBPS
*
```

Sample Network II.5 Listing

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
*DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL
***** NETIN RELEASE 7.02  THIS FILE SAVED 12/16/1991 08:45:42

***** GLOBAL VARIABLES
GLOBAL FLAGS =
  ANTITHETIC VARIATE = NO
  RANDOMIZER =  0
  MINIMIZE RANDOM SEED ARRAY = NO
  NETIN TIME UNITS = SECONDS
  ITERATE BY PRIORITY = NO
  CLOCK = YES
  CLOCK INCREMENT = .002 SECONDS
  BATCH = YES
  INPUT LISTING = YES
  DEFAULT LISTING = NO
  LENGTH = 0.078 SECONDS
  RESET STATISTICS = 0.010 SECONDS
  RUNTIME WARNINGS = TERMINAL
  PLOT DATA FILE = NO
  WIDE REPORTS = NO
  TRACE = NO

***** SEMAPHORES
SOFTWARE TYPE = SEMAPHORE
  NAME = TICK.1A
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.2A
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.3A
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.4A
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.1B
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.2B
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.3B
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999
  NAME = TICK.4B
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES =         999

***** STATISTICAL DISTRIBUTION FUNCTIONS
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
 51  STATISTICAL DISTRIBUTIONS =
 52    NAME = PKTLEN
 53      TYPE = PATTERN
 54      TABLE =
             VALUE =          1000.000
 57    NAME = PACKET ST
 58      TYPE = PATTERN
 59      TABLE =
             VALUE =             0.
 62    NAME = G2 PACKET IAT
 63      TYPE = PATTERN
 64      TABLE =
             VALUE =            33.333
 67    NAME = G3 PACKET IAT
 68      TYPE = PATTERN
 69      TABLE =
             VALUE =            25.000
 72    NAME = AUDIO PACKET IAT
 73      TYPE = PATTERN
 74      TABLE =
             VALUE =          1953.125
 77    NAME = VIDEO PACKET IAT
 78      TYPE = PATTERN
 79      TABLE =
             VALUE =            40.000
 82    NAME = FDDI.FM
 83      TYPE = PATTERN
 84      TABLE =
             VALUE =          1000.000
 87    NAME = FILL DATA
 88      TYPE = FILE.LINEAR
 89      A =         -1.000000
 90      B =      10200.000000
 91      LOWER.BOUND =           0.
 92      UPPER.BOUND =      10200.000000
 93    NAME = VALID DATA
 94      TYPE = FILE.LINEAR
 95      A =           1.000000
 96      B =           0.
 97      LOWER.BOUND =           0.
 98      UPPER.BOUND =      10200.000000
 99    NAME = RSCLEN
100      TYPE = PATTERN
101      TABLE =
             VALUE =          125.490
104    NAME = CRCLEN
105      TYPE = PATTERN
106      TABLE =
             VALUE =            6.275
109    NAME = HDRLEN
```

```
110       TYPE = PATTERN
111       TABLE =
              VALUE =            4.706
114
115   ***** PROCESSING ELEMENTS
116   HARDWARE TYPE = PROCESSING
117     NAME = G2A TRAFFIC
118       BASIC CYCLE TIME =         .010000 MIC
119       INPUT CONTROLLER = YES
120       INSTRUCTION REPERTOIRE =
121         INSTRUCTION TYPE = WRITE
122           NAME ; WRITE G2 DATA TO BUFFER
123             STORAGE DEVICE TO ACCESS ; GRADE 2 BUFFER
124             FILE ACCESSED ; G2 ASYNC DATA
125             NUMBER OF BITS TO TRANSMIT ; PKTLEN
126             REPLACE FLAG ; NO
127             RESUME FLAG ; NO
128             ALLOWABLE BUSSES ;
129               G2A BUS
130         INSTRUCTION TYPE = MESSAGE
131           NAME ; SEND G2 DATA PACKET
132             MESSAGE ; PKT_G2A
133             LENGTH ; PKTLEN
134             DESTINATION PROCESSOR ; FDDI 1A
135             QUEUE FLAG ; NO
136             RESUME FLAG ; NO
137             ALLOWABLE BUSSES ;
138               G2A BUS
139     NAME = G2B TRAFFIC
140       BASIC CYCLE TIME =         .010000 MIC
141       INPUT CONTROLLER = YES
142       INSTRUCTION REPERTOIRE =
143         INSTRUCTION TYPE = WRITE
144           NAME ; WRITE G2 DATA TO BUFFER
145             STORAGE DEVICE TO ACCESS ; GRADE 2 BUFFER
146             FILE ACCESSED ; G2 ASYNC DATA
147             NUMBER OF BITS TO TRANSMIT ; PKTLEN
148             REPLACE FLAG ; NO
149             RESUME FLAG ; NO
150             ALLOWABLE BUSSES ;
151               G2B BUS
152         INSTRUCTION TYPE = MESSAGE
153           NAME ; SEND G2 DATA PACKET
154             MESSAGE ; PKT_G2B
155             LENGTH ; PKTLEN
156             DESTINATION PROCESSOR ; FDDI 1B
157             QUEUE FLAG ; NO
158             RESUME FLAG ; NO
159             ALLOWABLE BUSSES ;
160               G2B BUS
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
161    NAME = G3A TRAFFIC
162      BASIC CYCLE TIME =        .010000 MIC
163      INPUT CONTROLLER = YES
164      INSTRUCTION REPERTOIRE =
165        INSTRUCTION TYPE = WRITE
166          NAME ; WRITE G3 DATA TO BUFFER
167            STORAGE DEVICE TO ACCESS ; GRADE 3 BUFFER
168            FILE ACCESSED ; G3 ASYNC DATA
169            NUMBER OF BITS TO TRANSMIT ; PKTLEN
170            REPLACE FLAG ; NO
171            RESUME FLAG ; NO
172            ALLOWABLE BUSSES ;
173              G3A BUS
174        INSTRUCTION TYPE = MESSAGE
175          NAME ; SEND G3 DATA PACKET
176            MESSAGE ; PKT_G3A
177            LENGTH ; PKTLEN
178            DESTINATION PROCESSOR ; FDDI 2A
179            QUEUE FLAG ; NO
180            RESUME FLAG ; NO
181            ALLOWABLE BUSSES ;
182              G3A BUS
183    NAME = G3B TRAFFIC
184      BASIC CYCLE TIME =        .010000 MIC
185      INPUT CONTROLLER = YES
186      INSTRUCTION REPERTOIRE =
187        INSTRUCTION TYPE = WRITE
188          NAME ; WRITE G3 DATA TO BUFFER
189            STORAGE DEVICE TO ACCESS ; GRADE 3 BUFFER
190            FILE ACCESSED ; G3 ASYNC DATA
191            NUMBER OF BITS TO TRANSMIT ; PKTLEN
192            REPLACE FLAG ; NO
193            RESUME FLAG ; NO
194            ALLOWABLE BUSSES ;
195              G3B BUS
196        INSTRUCTION TYPE = MESSAGE
197          NAME ; SEND G3 DATA PACKET
198            MESSAGE ; PKT_G3B
199            LENGTH ; PKTLEN
200            DESTINATION PROCESSOR ; FDDI 2B
201            QUEUE FLAG ; NO
202            RESUME FLAG ; NO
203            ALLOWABLE BUSSES ;
204              G3B BUS
205    NAME = AUDIO DATA A
206      BASIC CYCLE TIME =        .010000 MIC
207      INPUT CONTROLLER = YES
208      INSTRUCTION REPERTOIRE =
209        INSTRUCTION TYPE = MESSAGE
210          NAME ; SEND AUDIO DATA PACKET
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
211             MESSAGE ; PKT.AUDA
212             LENGTH ; PKTLEN
213             DESTINATION PROCESSOR ; FDDI 3A
214             QUEUE FLAG ; YES
215             RESUME FLAG ; NO
216             ALLOWABLE BUSSES ;
217               FDDI LAN A
218           INSTRUCTION TYPE = SEMAPHORE
219             NAME ; SUBTRACT 1 FROM TICK.4A
220               SEMAPHORE ; TICK.4A
221               SET/RESET FLAG ; RESET
222       NAME = AUDIO DATA B
223         BASIC CYCLE TIME =        .010000 MIC
224         INPUT CONTROLLER = YES
225         INSTRUCTION REPERTOIRE =
226           INSTRUCTION TYPE = MESSAGE
227             NAME ; SEND AUDIO DATA PACKET
228               MESSAGE ; PKT.AUDB
229               LENGTH ; PKTLEN
230               DESTINATION PROCESSOR ; FDDI 3B
231               QUEUE FLAG ; YES
232               RESUME FLAG ; NO
233               ALLOWABLE BUSSES ;
234                 FDDI LAN B
235           INSTRUCTION TYPE = SEMAPHORE
236             NAME ; SUBTRACT 1 FROM TICK.4B
237               SEMAPHORE ; TICK.4B
238               SET/RESET FLAG ; RESET
239       NAME = VIDEO DATA
240         BASIC CYCLE TIME =        .010000 MIC
241         INPUT CONTROLLER = YES
242         INSTRUCTION REPERTOIRE =
243           INSTRUCTION TYPE = MESSAGE
244             NAME ; SEND VIDEO DATA PACKET
245               MESSAGE ; PKT_VID
246               LENGTH ; PKTLEN
247               DESTINATION PROCESSOR ; CADU GEN
248               QUEUE FLAG ; NO
249               RESUME FLAG ; NO
250               ALLOWABLE BUSSES ;
251                 CADU GEN BUS
252       NAME = FDDI 1A
253         BASIC CYCLE TIME =        .010000 MIC
254         INPUT CONTROLLER = YES
255         INSTRUCTION REPERTOIRE =
256           INSTRUCTION TYPE = READ
257             NAME ; READ G2 DATA FROM BUFFER
258               STORAGE DEVICE TO ACCESS ; GRADE 2 BUFFER
259               FILE ACCESSED ; G2 ASYNC DATA
260               NUMBER OF BITS TO TRANSMIT ; FDDI.FM
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
261             DESTROY FLAG ; YES
262             PARTIAL FLAG ; YES
263             RESUME FLAG ; NO
264             ALLOWABLE BUSSES ;
265               G2A BUS
266           INSTRUCTION TYPE = MESSAGE
267             NAME ; SEND DATA PACKET ON FDDI
268               MESSAGE ; GS2.1A
269               LENGTH ; PKTLEN
270               DESTINATION PROCESSOR ; FDDI 3A
271               QUEUE FLAG ; YES
272               RESUME FLAG ; NO
273               ALLOWABLE BUSSES ;
274                 FDDI LAN A
275           INSTRUCTION TYPE = SEMAPHORE
276             NAME ; SUBTRACT 1 FROM TICK.1A
277               SEMAPHORE ; TICK.1A
278               SET/RESET FLAG ; RESET
279       NAME = FDDI 1B
280         BASIC CYCLE TIME =        .010000 MIC
281         INPUT CONTROLLER = YES
282         INSTRUCTION REPERTOIRE =
283           INSTRUCTION TYPE = READ
284             NAME ; READ G2 DATA FROM BUFFER
285               STORAGE DEVICE TO ACCESS ; GRADE 2 BUFFER
286               FILE ACCESSED ; G2 ASYNC DATA
287               NUMBER OF BITS TO TRANSMIT ; FDDI.FM
288               DESTROY FLAG ; YES
289               PARTIAL FLAG ; YES
290               RESUME FLAG ; NO
291               ALLOWABLE BUSSES ;
292                 G2B BUS
293           INSTRUCTION TYPE = MESSAGE
294             NAME ; SEND DATA PACKET ON FDDI
295               MESSAGE ; GS2.1B
296               LENGTH ; PKTLEN
297               DESTINATION PROCESSOR ; FDDI 3B
298               QUEUE FLAG ; YES
299               RESUME FLAG ; NO
300               ALLOWABLE BUSSES ;
301                 FDDI LAN B
302           INSTRUCTION TYPE = SEMAPHORE
303             NAME ; SUBTRACT 1 FROM TICK.1B
304               SEMAPHORE ; TICK.1B
305               SET/RESET FLAG ; RESET
306       NAME = FDDI 2A
307         BASIC CYCLE TIME =        .010000 MIC
308         INPUT CONTROLLER = YES
309         INSTRUCTION REPERTOIRE =
310           INSTRUCTION TYPE = READ
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
311           NAME ; READ G3 DATA FROM BUFFER
312             STORAGE DEVICE TO ACCESS ; GRADE 3 BUFFER
313             FILE ACCESSED ; G3 ASYNC DATA
314             NUMBER OF BITS TO TRANSMIT ; FDDI.FM
315             DESTROY FLAG ; YES
316             PARTIAL FLAG ; YES
317             RESUME FLAG ; NO
318             ALLOWABLE BUSSES ;
319               G3A BUS
320         INSTRUCTION TYPE = MESSAGE
321           NAME ; SEND DATA PACKET ON FDDI
322             MESSAGE ; GS3.2A
323             LENGTH ; PKTLEN
324             DESTINATION PROCESSOR ; FDDI 3A
325             QUEUE FLAG ; YES
326             RESUME FLAG ; NO
327             ALLOWABLE BUSSES ;
328               FDDI LAN A
329         INSTRUCTION TYPE = SEMAPHORE
330           NAME ; SUBTRACT 1 FROM TICK.2A
331             SEMAPHORE ; TICK.2A
332             SET/RESET FLAG ; RESET
333     NAME = FDDI 2B
334       BASIC CYCLE TIME =        .010000 MIC
335       INPUT CONTROLLER = YES
336       INSTRUCTION REPERTOIRE =
337         INSTRUCTION TYPE = READ
338           NAME ; READ G3 DATA FROM BUFFER
339             STORAGE DEVICE TO ACCESS ; GRADE 3 BUFFER
340             FILE ACCESSED ; G3 ASYNC DATA
341             NUMBER OF BITS TO TRANSMIT ; FDDI.FM
342             DESTROY FLAG ; YES
343             PARTIAL FLAG ; YES
344             RESUME FLAG ; NO
345             ALLOWABLE BUSSES ;
346               G3B BUS
347         INSTRUCTION TYPE = MESSAGE
348           NAME ; SEND DATA PACKET ON FDDI
349             MESSAGE ; GS3.2B
350             LENGTH ; PKTLEN
351             DESTINATION PROCESSOR ; FDDI 3B
352             QUEUE FLAG ; YES
353             RESUME FLAG ; NO
354             ALLOWABLE BUSSES ;
355               FDDI LAN B
356         INSTRUCTION TYPE = SEMAPHORE
357           NAME ; SUBTRACT 1 FROM TICK.2B
358             SEMAPHORE ; TICK.2B
359             SET/RESET FLAG ; RESET
360     NAME = FDDI 3A
```

```
361       BASIC CYCLE TIME =        .010000 MIC
362       INPUT CONTROLLER = YES
363       INSTRUCTION REPERTOIRE =
364         INSTRUCTION TYPE = WRITE
365           NAME ; WRITE ASYNC DATA TO BUFFER
366             STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
367             FILE ACCESSED ; ASYNC DATA
368             NUMBER OF BITS TO TRANSMIT ; FDDI.FM
369             REPLACE FLAG ; NO
370             RESUME FLAG ; NO
371             ALLOWABLE BUSSES ;
372               CADU GEN BUS
373         INSTRUCTION TYPE = WRITE
374           NAME ; WRITE HEADER DATA TO BUFFER
375             STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
376             FILE ACCESSED ; ASYNC DATA
377             NUMBER OF BITS TO TRANSMIT ; HDRLEN
378             REPLACE FLAG ; NO
379             RESUME FLAG ; NO
380             ALLOWABLE BUSSES ;
381               CADU GEN BUS
382         INSTRUCTION TYPE = MESSAGE
383           NAME ; SEND DATA PACKET TO CADU GEN
384             MESSAGE ; PKT.THREE
385             LENGTH ; PKTLEN
386             DESTINATION PROCESSOR ; CADU GEN
387             QUEUE FLAG ; NO
388             RESUME FLAG ; NO
389             ALLOWABLE BUSSES ;
390               CADU GEN BUS
391           NAME ; SEND G2 DATA TO RS CODE GEN
392             MESSAGE ; RSC
393             LENGTH ; 10 BITS
394             DESTINATION PROCESSOR ; RS CODE GEN
395             QUEUE FLAG ; NO
396             RESUME FLAG ; NO
397             ALLOWABLE BUSSES ;
398               CADU GEN BUS
399           NAME ; SEND G3 DATA TO CRC CODE GEN
400             MESSAGE ; CRC
401             LENGTH ; 10 BITS
402             DESTINATION PROCESSOR ; CRC CODE GEN
403             QUEUE FLAG ; NO
404             RESUME FLAG ; NO
405             ALLOWABLE BUSSES ;
406               CADU GEN BUS
407         INSTRUCTION TYPE = SEMAPHORE
408           NAME ; SUBTRACT 1 FROM TICK.3A
409             SEMAPHORE ; TICK.3A
410             SET/RESET FLAG ; RESET
```

```
411     NAME = FDDI 3B
412       BASIC CYCLE TIME =        .010000 MIC
413       INPUT CONTROLLER = YES
414       INSTRUCTION REPERTOIRE =
415         INSTRUCTION TYPE = WRITE
416           NAME ; WRITE ASYNC DATA TO BUFFER
417             STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
418             FILE ACCESSED ; ASYNC DATA
419             NUMBER OF BITS TO TRANSMIT ; FDDI.FM
420             REPLACE FLAG ; NO
421             RESUME FLAG ; NO
422             ALLOWABLE BUSSES ;
423               CADU GEN BUS
424         INSTRUCTION TYPE = WRITE
425           NAME ; WRITE HEADER DATA TO BUFFER
426             STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
427             FILE ACCESSED ; ASYNC DATA
428             NUMBER OF BITS TO TRANSMIT ; HDRLEN
429             REPLACE FLAG ; NO
430             RESUME FLAG ; NO
431             ALLOWABLE BUSSES ;
432               CADU GEN BUS
433         INSTRUCTION TYPE = MESSAGE
434           NAME ; SEND DATA PACKET TO CADU GEN
435             MESSAGE ; PKT.THREE
436             LENGTH ; PKTLEN
437             DESTINATION PROCESSOR ; CADU GEN
438             QUEUE FLAG ; NO
439             RESUME FLAG ; NO
440             ALLOWABLE BUSSES ;
441               CADU GEN BUS
442           NAME ; SEND G2 DATA TO RS CODE GEN
443             MESSAGE ; RSC
444             LENGTH ; 10 BITS
445             DESTINATION PROCESSOR ; RS CODE GEN
446             QUEUE FLAG ; NO
447             RESUME FLAG ; NO
448             ALLOWABLE BUSSES ;
449               CADU GEN BUS
450           NAME ; SEND G3 DATA TO CRC CODE GEN
451             MESSAGE ; CRC
452             LENGTH ; 10 BITS
453             DESTINATION PROCESSOR ; CRC CODE GEN
454             QUEUE FLAG ; NO
455             RESUME FLAG ; NO
456             ALLOWABLE BUSSES ;
457               CADU GEN BUS
458         INSTRUCTION TYPE = SEMAPHORE
459           NAME ; SUBTRACT 1 FROM TICK.3B
460             SEMAPHORE ; TICK.3B
```

```
461             SET/RESET FLAG ; RESET
462       NAME = RS CODE GEN
463         BASIC CYCLE TIME =        .010000 MIC
464         INPUT CONTROLLER = YES
465         INSTRUCTION REPERTOIRE =
466           INSTRUCTION TYPE = WRITE
467             NAME ; WRITE RS CODE BITS TO BUFFER
468               STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
469               FILE ACCESSED ; ASYNC DATA
470               NUMBER OF BITS TO TRANSMIT ; RSCLEN
471               REPLACE FLAG ; NO
472               RESUME FLAG ; NO
473               ALLOWABLE BUSSES ;
474                 CADU GEN BUS
475       NAME = CRC CODE GEN
476         BASIC CYCLE TIME =        .010000 MIC
477         INPUT CONTROLLER = YES
478         INSTRUCTION REPERTOIRE =
479           INSTRUCTION TYPE = WRITE
480             NAME ; WRITE CRC CODE BITS TO BUFFER
481               STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
482               FILE ACCESSED ; ASYNC DATA
483               NUMBER OF BITS TO TRANSMIT ; CRCLEN
484               REPLACE FLAG ; NO
485               RESUME FLAG ; NO
486               ALLOWABLE BUSSES ;
487                 CADU GEN BUS
488       NAME = CHANNEL
489         BASIC CYCLE TIME =        .010000 MIC
490         INPUT CONTROLLER = YES
491         INSTRUCTION REPERTOIRE =
492           INSTRUCTION TYPE = PROCESSING
493             NAME ; SEND CHANNEL ACCESS DATA UNIT
494               TIME ; 1 CYCLES
495       NAME = CADU GEN
496         BASIC CYCLE TIME =       0.        MIC
497         INPUT CONTROLLER = YES
498         INSTRUCTION REPERTOIRE =
499           INSTRUCTION TYPE = READ
500             NAME ; READ ASYNC DATA FROM BUFFER
501               STORAGE DEVICE TO ACCESS ; FDDI 3 BUFFER
502               FILE ACCESSED ; ASYNC DATA
503               DESTROY FLAG ; YES
504               PARTIAL FLAG ; YES
505               RESUME FLAG ; NO
506               ALLOWABLE BUSSES ;
507                 CADU GEN BUS
508             NAME ; READ PCA.PDU FROM BUFFER
509               STORAGE DEVICE TO ACCESS ; VCDU DZ BUFFER
510               FILE ACCESSED ; CHANNEL ACCESS SLOT
```

```
511             NUMBER OF BITS TO TRANSMIT ; 10200
512             DESTROY FLAG ; YES
513             PARTIAL FLAG ; YES
514             RESUME FLAG ; NO
515             ALLOWABLE BUSSES ;
516               CADU GEN BUS
517           INSTRUCTION TYPE = WRITE
518             NAME ; WRITE FILL DATA TO BUFFER
519               STORAGE DEVICE TO ACCESS ; VCDU DZ BUFFER
520               FILE ACCESSED ; CHANNEL ACCESS SLOT
521               NUMBER OF BITS TO TRANSMIT ; FILL DATA
522               REPLACE FLAG ; NO
523               RESUME FLAG ; NO
524               ALLOWABLE BUSSES ;
525                 CADU GEN BUS
526             NAME ; WRITE VALID DATA TO BUFFER
527               STORAGE DEVICE TO ACCESS ; VCDU DZ BUFFER
528               FILE ACCESSED ; CHANNEL ACCESS SLOT
529               NUMBER OF BITS TO TRANSMIT ; VALID DATA
530               REPLACE FLAG ; NO
531               RESUME FLAG ; NO
532               ALLOWABLE BUSSES ;
533                 CADU GEN BUS
534           INSTRUCTION TYPE = MESSAGE
535             NAME ; GENERATE CADU PDU
536               MESSAGE ; CADU.PDU
537               LENGTH ; 10200 BITS
538               DESTINATION PROCESSOR ; CHANNEL
539               QUEUE FLAG ; NO
540               RESUME FLAG ; NO
541               ALLOWABLE BUSSES ;
542                 CADU GEN BUS
543       NAME = PHANTOM STATION A
544         BASIC CYCLE TIME =      0.       MIC
545         INPUT CONTROLLER = YES
546         INSTRUCTION REPERTOIRE =
547           INSTRUCTION TYPE = SEMAPHORE
548             NAME ; ADD 1 TO TICK.1A
549               SEMAPHORE ; TICK.1A
550               SET/RESET FLAG ; SET
551             NAME ; ADD 1 TO TICK.2A
552               SEMAPHORE ; TICK.2A
553               SET/RESET FLAG ; SET
554             NAME ; ADD 1 TO TICK.3A
555               SEMAPHORE ; TICK.3A
556               SET/RESET FLAG ; SET
557             NAME ; ADD 1 TO TICK.4A
558               SEMAPHORE ; TICK.4A
559               SET/RESET FLAG ; SET
560       NAME = PHANTOM STATION B
```

```
561       BASIC CYCLE TIME =       0.        MIC
562       INPUT CONTROLLER = YES
563       INSTRUCTION REPERTOIRE =
564         INSTRUCTION TYPE = SEMAPHORE
565           NAME ; ADD 1 TO TICK.1B
566             SEMAPHORE ; TICK.1B
567             SET/RESET FLAG ; SET
568           NAME ; ADD 1 TO TICK.2B
569             SEMAPHORE ; TICK.2B
570             SET/RESET FLAG ; SET
571           NAME ; ADD 1 TO TICK.3B
572             SEMAPHORE ; TICK.3B
573             SET/RESET FLAG ; SET
574           NAME ; ADD 1 TO TICK.4B
575             SEMAPHORE ; TICK.4B
576             SET/RESET FLAG ; SET
577
578   ***** TRANSFER DEVICES
579   HARDWARE TYPE = DATA TRANSFER
580     NAME = G2A BUS
581       CYCLE TIME =       0.        MIC
582       BITS PER CYCLE =          1
583       CYCLES PER WORD =         8
584       WORDS PER BLOCK =      1024
585       WORD OVERHEAD TIME =       .001000 MIC
586       BLOCK OVERHEAD TIME =      0.        MIC
587       PROTOCOL = FIRST COME FIRST SERVED
588       BUS CONNECTIONS =
589         G2A TRAFFIC
590         GRADE 2 BUFFER
591         FDDI 1A
592     NAME = G2B BUS
593       CYCLE TIME =       0.        MIC
594       BITS PER CYCLE =          1
595       CYCLES PER WORD =         8
596       WORDS PER BLOCK =      1024
597       WORD OVERHEAD TIME =       .001000 MIC
598       BLOCK OVERHEAD TIME =      0.        MIC
599       PROTOCOL = FIRST COME FIRST SERVED
600       BUS CONNECTIONS =
601         G2B TRAFFIC
602         GRADE 2 BUFFER
603         FDDI 1B
604     NAME = G3A BUS
605       CYCLE TIME =       0.        MIC
606       BITS PER CYCLE =          1
607       CYCLES PER WORD =         8
608       WORDS PER BLOCK =      1024
609       WORD OVERHEAD TIME =       .001000 MIC
610       BLOCK OVERHEAD TIME =      0.        MIC
```

```
611       PROTOCOL = FIRST COME FIRST SERVED
612       BUS CONNECTIONS =
613         G3A TRAFFIC
614         GRADE 3 BUFFER
615         FDDI 2A
616     NAME = G3B BUS
617       CYCLE TIME =       0.        MIC
618       BITS PER CYCLE =          1
619       CYCLES PER WORD =         8
620       WORDS PER BLOCK =      1024
621       WORD OVERHEAD TIME =       .001000 MIC
622       BLOCK OVERHEAD TIME =      0.       MIC
623       PROTOCOL = FIRST COME FIRST SERVED
624       BUS CONNECTIONS =
625         G3B TRAFFIC
626         GRADE 3 BUFFER
627         FDDI 2B
628     NAME = CADU GEN BUS
629       CYCLE TIME =       0.        MIC
630       BITS PER CYCLE =          1
631       CYCLES PER WORD =         8
632       WORDS PER BLOCK =      1024
633       WORD OVERHEAD TIME =      0.       MIC
634       BLOCK OVERHEAD TIME =     0.       MIC
635       PROTOCOL = FIRST COME FIRST SERVED
636       BUS CONNECTIONS =
637         VIDEO DATA
638         FDDI 3A
639         FDDI 3B
640         FDDI 3 BUFFER
641         VCDU DZ BUFFER
642         RS CODE GEN
643         CRC CODE GEN
644         CADU GEN
645         CHANNEL
646     NAME = FDDI LAN A
647       CYCLE TIME =       .008000 MIC
648       BITS PER CYCLE =          1
649       CYCLES PER WORD =         4
650       WORDS PER BLOCK =      3200
651       WORD OVERHEAD TIME =       .008000 MIC
652       BLOCK OVERHEAD TIME =      0.       MIC
653       PROTOCOL = PRIORITY TOKEN RING
654       TOKEN PASSING TIME = 0.1
655       BUS CONNECTIONS =
656         FDDI 1A
657           KEY = 1.0
658           SYNCHRONOUS = 200.0 MIC
659           MINIMUM SYNCHRONOUS PRIORITY =        7
660           TARGET TOKEN ROTATION TIMES =
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
661             PRIORITY       4 = 2500.0 MIC
662             PRIORITY       1 = 2500.0 MIC
663           FDDI 2A
664             KEY = 2.0
665             SYNCHRONOUS = 200.0 MIC
666             MINIMUM SYNCHRONOUS PRIORITY =          7
667             TARGET TOKEN ROTATION TIMES =
668               PRIORITY       4 = 2500.0 MIC
669               PRIORITY       1 = 2500.0 MIC
670           FDDI 3A
671             KEY = 3.0
672             SYNCHRONOUS = 200.0 MIC
673             MINIMUM SYNCHRONOUS PRIORITY =          7
674             TARGET TOKEN ROTATION TIMES =
675               PRIORITY       4 = 2500.0 MIC
676               PRIORITY       1 = 2500.0 MIC
677           AUDIO DATA A
678             KEY = 4.0
679             SYNCHRONOUS = 200.0 MIC
680             MINIMUM SYNCHRONOUS PRIORITY =          7
681             TARGET TOKEN ROTATION TIMES =
682               PRIORITY       4 = 2500.0 MIC
683               PRIORITY       1 = 2500.0 MIC
684       NAME = FDDI LAN B
685         CYCLE TIME =         .008000 MIC
686         BITS PER CYCLE =            1
687         CYCLES PER WORD =           4
688         WORDS PER BLOCK =        3200
689         WORD OVERHEAD TIME =         .008000 MIC
690         BLOCK OVERHEAD TIME =        0.        MIC
691         PROTOCOL = PRIORITY TOKEN RING
692         TOKEN PASSING TIME = 0.1
693         BUS CONNECTIONS =
694           FDDI 1B
695             KEY = 1.0
696             SYNCHRONOUS = 200.0 MIC
697             MINIMUM SYNCHRONOUS PRIORITY =          7
698             TARGET TOKEN ROTATION TIMES =
699               PRIORITY       4 = 2500.0 MIC
700               PRIORITY       1 = 2500.0 MIC
701           FDDI 2B
702             KEY = 2.0
703             SYNCHRONOUS = 200.0 MIC
704             MINIMUM SYNCHRONOUS PRIORITY =          7
705             TARGET TOKEN ROTATION TIMES =
706               PRIORITY       4 = 2500.0 MIC
707               PRIORITY       1 = 2500.0 MIC
708           FDDI 3B
709             KEY = 3.0
710             SYNCHRONOUS = 200.0 MIC
```

DUAL FDDI LAN SPACE DATA COMMUNICATIONS SYSTEM MODEL

```
711           MINIMUM SYNCHRONOUS PRIORITY =          7
712           TARGET TOKEN ROTATION TIMES =
713             PRIORITY       4 = 2500.0 MIC
714             PRIORITY       1 = 2500.0 MIC
715         AUDIO DATA B
716           KEY = 4.0
717           SYNCHRONOUS = 200.0 MIC
718           MINIMUM SYNCHRONOUS PRIORITY =          7
719           TARGET TOKEN ROTATION TIMES =
720             PRIORITY       4 = 2500.0 MIC
721             PRIORITY       1 = 2500.0 MIC
722
723  ***** STORAGE DEVICES
724  HARDWARE TYPE = STORAGE
725    NAME = GRADE 2 BUFFER
726      WORD ACCESS TIME = 0.0 MIC
727      BITS PER WORD =           8
728      WORDS PER BLOCK =        1024
729      READ OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
730      WRITE OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
731      OVERHEAD TIME PER BLOCK ACCESS = 0.0 MIC
732      CAPACITY =     100000000000. BITS
733      NUMBER OF PORTS =           2
734    NAME = GRADE 3 BUFFER
735      WORD ACCESS TIME = 0.0 MIC
736      BITS PER WORD =           8
737      WORDS PER BLOCK =        1024
738      READ OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
739      WRITE OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
740      OVERHEAD TIME PER BLOCK ACCESS = 0.0 MIC
741      CAPACITY =     100000000000. BITS
742      NUMBER OF PORTS =           2
743    NAME = FDDI 3 BUFFER
744      WORD ACCESS TIME = 0.0 MIC
745      BITS PER WORD =           8
746      WORDS PER BLOCK =        1024
747      READ OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
748      WRITE OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
749      OVERHEAD TIME PER BLOCK ACCESS = 0.0 MIC
750      CAPACITY =         10000000. BITS
751      NUMBER OF PORTS =           2
752    NAME = VCDU DZ BUFFER
753      WORD ACCESS TIME = 0.0 MIC
754      BITS PER WORD =           8
755      WORDS PER BLOCK =        1024
756      READ OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
757      WRITE OVERHEAD TIME PER WORD ACCESS = 0.0 MIC
758      OVERHEAD TIME PER BLOCK ACCESS = 0.0 MIC
759      CAPACITY =           10200. BITS
760      NUMBER OF PORTS =           2
```

```
761
762  ***** MODULES
763  SOFTWARE TYPE = MODULE
764    NAME = FDDI 1A MODULE
765      PRIORITY =          4
766      INTERRUPTIBILITY FLAG = NO
767      CONCURRENT EXECUTION = NO
768      START TIME = PACKET ST
769      ALLOWED PROCESSORS =
770        FDDI 1A
771      REQUIRED MESSAGES =
772        PKT_G2A
773      REQUIRED SEMAPHORE STATUS =
774        RUN WHEN ; TICK.1A
775        IS ; > 0
776      INSTRUCTION LIST =
777        EXECUTE A TOTAL OF ; 1 SEND FDDI 1A FRAME
778      ANDED SUCCESSORS =
779        CHAIN TO ; FDDI 1A MODULE
780        WITH ITERATIONS THEN CHAIN COUNT OF ;      0
781    NAME = FDDI 1B MODULE
782      PRIORITY =          4
783      INTERRUPTIBILITY FLAG = NO
784      CONCURRENT EXECUTION = NO
785      START TIME = PACKET ST
786      ALLOWED PROCESSORS =
787        FDDI 1B
788      REQUIRED MESSAGES =
789        PKT_G2B
790      REQUIRED SEMAPHORE STATUS =
791        RUN WHEN ; TICK.1B
792        IS ; > 0
793      INSTRUCTION LIST =
794        EXECUTE A TOTAL OF ; 1 SEND FDDI 1B FRAME
795      ANDED SUCCESSORS =
796        CHAIN TO ; FDDI 1B MODULE
797        WITH ITERATIONS THEN CHAIN COUNT OF ;      0
798    NAME = FDDI 2A MODULE
799      PRIORITY =          1
800      INTERRUPTIBILITY FLAG = NO
801      CONCURRENT EXECUTION = NO
802      START TIME = PACKET ST
803      ALLOWED PROCESSORS =
804        FDDI 2A
805      REQUIRED MESSAGES =
806        PKT_G3A
807      REQUIRED SEMAPHORE STATUS =
808        RUN WHEN ; TICK.2A
809        IS ; > 0
810      INSTRUCTION LIST =
```

```
811         EXECUTE A TOTAL OF ; 1 SEND FDDI 2A FRAME
812       ANDED SUCCESSORS =
813         CHAIN TO ; FDDI 2A MODULE
814         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
815     NAME = FDDI 2B MODULE
816       PRIORITY =          1
817       INTERRUPTIBILITY FLAG = NO
818       CONCURRENT EXECUTION = NO
819       START TIME = PACKET ST
820       ALLOWED PROCESSORS =
821         FDDI 2B
822       REQUIRED MESSAGES =
823         PKT_G3B
824       REQUIRED SEMAPHORE STATUS =
825         RUN WHEN ; TICK.2B
826         IS ; > 0
827       INSTRUCTION LIST =
828         EXECUTE A TOTAL OF ; 1 SEND FDDI 2B FRAME
829       ANDED SUCCESSORS =
830         CHAIN TO ; FDDI 2B MODULE
831         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
832     NAME = FDDI 3A AUD MODULE
833       PRIORITY =          7
834       INTERRUPTIBILITY FLAG = NO
835       CONCURRENT EXECUTION = NO
836       START TIME = PACKET ST
837       ALLOWED PROCESSORS =
838         FDDI 3A
839       REQUIRED MESSAGES =
840         PKT.AUDA
841       REQUIRED SEMAPHORE STATUS =
842         RUN WHEN ; TICK.3A
843         IS ; > 0
844       INSTRUCTION LIST =
845         EXECUTE A TOTAL OF ; 1 RECEIVE AUDIO FRAME
846         EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.3A
847       ANDED SUCCESSORS =
848         CHAIN TO ; FDDI 3A AUD MODULE
849         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
850     NAME = FDDI 3B AUD MODULE
851       PRIORITY =          7
852       INTERRUPTIBILITY FLAG = NO
853       CONCURRENT EXECUTION = NO
854       START TIME = PACKET ST
855       ALLOWED PROCESSORS =
856         FDDI 3B
857       REQUIRED MESSAGES =
858         PKT.AUDB
859       REQUIRED SEMAPHORE STATUS =
860         RUN WHEN ; TICK.3B
```

```
861         IS ; > 0
862       INSTRUCTION LIST =
863         EXECUTE A TOTAL OF ; 1 RECEIVE AUDIO FRAME
864         EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.3B
865       ANDED SUCCESSORS =
866         CHAIN TO ; FDDI 3B AUD MODULE
867         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
868     NAME = FDDI 3A G2 MODULE
869       PRIORITY =           4
870       INTERRUPTIBILITY FLAG = NO
871       CONCURRENT EXECUTION = NO
872       START TIME = PACKET ST
873       ALLOWED PROCESSORS =
874         FDDI 3A
875       REQUIRED MESSAGES =
876         GS2.1A
877       REQUIRED SEMAPHORE STATUS =
878         RUN WHEN ; TICK.3A
879         IS ; > 0
880       INSTRUCTION LIST =
881         EXECUTE A TOTAL OF ; 1 RECEIVE G2 DATA FRAME
882         EXECUTE A TOTAL OF ; 1 WRITE HEADER DATA TO BUFFER
883         EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.3A
884       ANDED SUCCESSORS =
885         CHAIN TO ; FDDI 3A G2 MODULE
886         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
887     NAME = FDDI 3B G2 MODULE
888       PRIORITY =           4
889       INTERRUPTIBILITY FLAG = NO
890       CONCURRENT EXECUTION = NO
891       START TIME = PACKET ST
892       ALLOWED PROCESSORS =
893         FDDI 3B
894       REQUIRED MESSAGES =
895         GS2.1B
896       REQUIRED SEMAPHORE STATUS =
897         RUN WHEN ; TICK.3B
898         IS ; > 0
899       INSTRUCTION LIST =
900         EXECUTE A TOTAL OF ; 1 RECEIVE G2 DATA FRAME
901         EXECUTE A TOTAL OF ; 1 WRITE HEADER DATA TO BUFFER
902         EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.3B
903       ANDED SUCCESSORS =
904         CHAIN TO ; FDDI 3B G2 MODULE
905         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
906     NAME = FDDI 3A G3 MODULE
907       PRIORITY =           1
908       INTERRUPTIBILITY FLAG = NO
909       CONCURRENT EXECUTION = NO
910       START TIME = PACKET ST
```

```
911       ALLOWED PROCESSORS =
912         FDDI 3A
913       REQUIRED MESSAGES =
914         GS3.2A
915       REQUIRED SEMAPHORE STATUS =
916         RUN WHEN ; TICK.3A
917         IS ; > 0
918       INSTRUCTION LIST =
919         EXECUTE A TOTAL OF ; 1 RECEIVE G3 DATA FRAME
920         EXECUTE A TOTAL OF ; 1 WRITE HEADER DATA TO BUFFER
921         EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.3A
922       ANDED SUCCESSORS =
923         CHAIN TO ; FDDI 3A G3 MODULE
924         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
925     NAME = FDDI 3B G3 MODULE
926       PRIORITY =           1
927       INTERRUPTIBILITY FLAG = NO
928       CONCURRENT EXECUTION = NO
929       START TIME = PACKET ST
930       ALLOWED PROCESSORS =
931         FDDI 3B
932       REQUIRED MESSAGES =
933         GS3.2B
934       REQUIRED SEMAPHORE STATUS =
935         RUN WHEN ; TICK.3B
936         IS ; > 0
937       INSTRUCTION LIST =
938         EXECUTE A TOTAL OF ; 1 RECEIVE G3 DATA FRAME
939         EXECUTE A TOTAL OF ; 1 WRITE HEADER DATA TO BUFFER
940         EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.3B
941       ANDED SUCCESSORS =
942         CHAIN TO ; FDDI 3B G3 MODULE
943         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
944     NAME = CRC CODE GEN MODULE
945       PRIORITY =           7
946       INTERRUPTIBILITY FLAG = NO
947       CONCURRENT EXECUTION = NO
948       START TIME = PACKET ST
949       ALLOWED PROCESSORS =
950         CRC CODE GEN
951       REQUIRED MESSAGES =
952         CRC
953       INSTRUCTION LIST =
954         EXECUTE A TOTAL OF ; 1 WRITE CRC CODE BITS TO BUFFER
955       ANDED SUCCESSORS =
956         CHAIN TO ; CRC CODE GEN MODULE
957         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
958     NAME = RS CODE GEN MODULE
959       PRIORITY =           7
960       INTERRUPTIBILITY FLAG = NO
```

```
  961       CONCURRENT EXECUTION = NO
  962       START TIME = PACKET ST
  963       ALLOWED PROCESSORS =
  964         RS CODE GEN
  965       REQUIRED MESSAGES =
  966         RSC
  967       INSTRUCTION LIST =
  968         EXECUTE A TOTAL OF ; 1 WRITE RS CODE BITS TO BUFFER
  969       ANDED SUCCESSORS =
  970         CHAIN TO ; RS CODE GEN MODULE
  971         WITH ITERATIONS THEN CHAIN COUNT OF ;       0
  972     NAME = G2A TRAFFIC MOD
  973       PRIORITY =          0
  974       INTERRUPTIBILITY FLAG = NO
  975       CONCURRENT EXECUTION = NO
  976       ITERATION PERIOD = G2 PACKET IAT
  977       START TIME = PACKET ST
  978       ALLOWED PROCESSORS =
  979         G2A TRAFFIC
  980       INSTRUCTION LIST =
  981         EXECUTE A TOTAL OF ; 1 WRITE G2 DATA TO BUFFER
  982         EXECUTE A TOTAL OF ; 1 SEND G2 DATA PACKET
  983     NAME = G2B TRAFFIC MOD
  984       PRIORITY =          0
  985       INTERRUPTIBILITY FLAG = NO
  986       CONCURRENT EXECUTION = NO
  987       ITERATION PERIOD = G2 PACKET IAT
  988       START TIME = PACKET ST
  989       ALLOWED PROCESSORS =
  990         G2B TRAFFIC
  991       INSTRUCTION LIST =
  992         EXECUTE A TOTAL OF ; 1 WRITE G2 DATA TO BUFFER
  993         EXECUTE A TOTAL OF ; 1 SEND G2 DATA PACKET
  994     NAME = G3A TRAFFIC MOD
  995       PRIORITY =          0
  996       INTERRUPTIBILITY FLAG = NO
  997       CONCURRENT EXECUTION = NO
  998       ITERATION PERIOD = G3 PACKET IAT
  999       START TIME = PACKET ST
 1000       ALLOWED PROCESSORS =
 1001         G3A TRAFFIC
 1002       INSTRUCTION LIST =
 1003         EXECUTE A TOTAL OF ; 1 WRITE G3 DATA TO BUFFER
 1004         EXECUTE A TOTAL OF ; 1 SEND G3 DATA PACKET
 1005     NAME = G3B TRAFFIC MOD
 1006       PRIORITY =          0
 1007       INTERRUPTIBILITY FLAG = NO
 1008       CONCURRENT EXECUTION = NO
 1009       ITERATION PERIOD = G3 PACKET IAT
 1010       START TIME = PACKET ST
```

```
1011      ALLOWED PROCESSORS =
1012        G3B TRAFFIC
1013      INSTRUCTION LIST =
1014        EXECUTE A TOTAL OF ; 1 WRITE G3 DATA TO BUFFER
1015        EXECUTE A TOTAL OF ; 1 SEND G3 DATA PACKET
1016    NAME = AUDIO DATA A MOD
1017      PRIORITY =          7
1018      INTERRUPTIBILITY FLAG = NO
1019      CONCURRENT EXECUTION = NO
1020      ITERATION PERIOD = AUDIO PACKET IAT
1021      START TIME = PACKET ST
1022      ALLOWED PROCESSORS =
1023        AUDIO DATA A
1024      REQUIRED SEMAPHORE STATUS =
1025        RUN WHEN ; TICK.4A
1026        IS ; > 0
1027      INSTRUCTION LIST =
1028        EXECUTE A TOTAL OF ; 1 SEND AUDIO DATA PACKET
1029        EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.4A
1030    NAME = AUDIO DATA B MOD
1031      PRIORITY =          7
1032      INTERRUPTIBILITY FLAG = NO
1033      CONCURRENT EXECUTION = NO
1034      ITERATION PERIOD = AUDIO PACKET IAT
1035      START TIME = PACKET ST
1036      ALLOWED PROCESSORS =
1037        AUDIO DATA B
1038      REQUIRED SEMAPHORE STATUS =
1039        RUN WHEN ; TICK.4B
1040        IS ; > 0
1041      INSTRUCTION LIST =
1042        EXECUTE A TOTAL OF ; 1 SEND AUDIO DATA PACKET
1043        EXECUTE A TOTAL OF ; 1 SUBTRACT 1 FROM TICK.4B
1044    NAME = VIDEO DATA MOD
1045      PRIORITY =          0
1046      INTERRUPTIBILITY FLAG = NO
1047      CONCURRENT EXECUTION = NO
1048      ITERATION PERIOD = VIDEO PACKET IAT
1049      START TIME = PACKET ST
1050      ALLOWED PROCESSORS =
1051        VIDEO DATA
1052      INSTRUCTION LIST =
1053        EXECUTE A TOTAL OF ; 1 SEND VIDEO DATA PACKET
1054    NAME = CHANNEL MOD
1055      PRIORITY =          0
1056      INTERRUPTIBILITY FLAG = NO
1057      CONCURRENT EXECUTION = NO
1058      START TIME = PACKET ST
1059      ALLOWED PROCESSORS =
1060        CHANNEL
```

```
1061      REQUIRED MESSAGES =
1062        CADU.PDU
1063      INSTRUCTION LIST =
1064        EXECUTE A TOTAL OF ; 1 SEND CHANNEL ACCESS DATA UNIT
1065    NAME = CADU GEN MOD
1066      PRIORITY =          0
1067      INTERRUPTIBILITY FLAG = NO
1068      CONCURRENT EXECUTION = NO
1069      ITERATION PERIOD = 34.0 MIC
1070      START TIME = PACKET ST
1071      ALLOWED PROCESSORS =
1072        CADU GEN
1073      INSTRUCTION LIST =
1074        EXECUTE A TOTAL OF ; 1 READ ASYNC DATA FROM BUFFER
1075        EXECUTE A TOTAL OF ; 1 WRITE FILL DATA TO BUFFER
1076        EXECUTE A TOTAL OF ; 1 WRITE VALID DATA TO BUFFER
1077        EXECUTE A TOTAL OF ; 1 READ PCA.PDU FROM BUFFER
1078        EXECUTE A TOTAL OF ; 1 GENERATE CADU PDU
1079    NAME = BROADCAST TICK A
1080      PRIORITY =          0
1081      INTERRUPTIBILITY FLAG = NO
1082      CONCURRENT EXECUTION = NO
1083      ITERATION PERIOD = 8.0 MIC
1084      START TIME = PACKET ST
1085      ALLOWED PROCESSORS =
1086        PHANTOM STATION A
1087      INSTRUCTION LIST =
1088        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.1A
1089        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.2A
1090        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.3A
1091        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.4A
1092    NAME = BROADCAST TICK B
1093      PRIORITY =          0
1094      INTERRUPTIBILITY FLAG = NO
1095      CONCURRENT EXECUTION = NO
1096      ITERATION PERIOD = 8.0 MIC
1097      START TIME = PACKET ST
1098      ALLOWED PROCESSORS =
1099        PHANTOM STATION B
1100      INSTRUCTION LIST =
1101        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.1B
1102        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.2B
1103        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.3B
1104        EXECUTE A TOTAL OF ; 1 ADD 1 TO TICK.4B
1105
1106  ***** FILES
1107  SOFTWARE TYPE = FILE
1108    NAME = G2 ASYNC DATA
1109      NUMBER OF BITS =            0.
1110      INITIAL RESIDENCY =
```

```
1111            GRADE 2 BUFFER
1112        READ ONLY FLAG = NO
1113      NAME = G3 ASYNC DATA
1114        NUMBER OF BITS =              0.
1115        INITIAL RESIDENCY =
1116            GRADE 3 BUFFER
1117        READ ONLY FLAG = NO
1118      NAME = ASYNC DATA
1119        NUMBER OF BITS =              0.
1120        INITIAL RESIDENCY =
1121            FDDI 3 BUFFER
1122        READ ONLY FLAG = NO
1123      NAME = CHANNEL ACCESS SLOT
1124        NUMBER OF BITS =              0.
1125        INITIAL RESIDENCY =
1126            VCDU DZ BUFFER
1127        READ ONLY FLAG = NO
1128
1129  ***** MACRO INSTRUCTIONS
1130  SOFTWARE TYPE = MACRO INSTRUCTION
1131    NAME = SEND FDDI 1A FRAME
1132      NUMBER OF INSTRUCTIONS ; 1
1133      INSTRUCTION NAME ; READ G2 DATA FROM BUFFER
1134      NUMBER OF INSTRUCTIONS ; 1
1135      INSTRUCTION NAME ; SEND DATA PACKET ON FDDI
1136      NUMBER OF INSTRUCTIONS ; 1
1137      INSTRUCTION NAME ; SUBTRACT 1 FROM TICK.1A
1138    NAME = SEND FDDI 1B FRAME
1139      NUMBER OF INSTRUCTIONS ; 1
1140      INSTRUCTION NAME ; READ G2 DATA FROM BUFFER
1141      NUMBER OF INSTRUCTIONS ; 1
1142      INSTRUCTION NAME ; SEND DATA PACKET ON FDDI
1143      NUMBER OF INSTRUCTIONS ; 1
1144      INSTRUCTION NAME ; SUBTRACT 1 FROM TICK.1B
1145    NAME = SEND FDDI 2A FRAME
1146      NUMBER OF INSTRUCTIONS ; 1
1147      INSTRUCTION NAME ; READ G3 DATA FROM BUFFER
1148      NUMBER OF INSTRUCTIONS ; 1
1149      INSTRUCTION NAME ; SEND DATA PACKET ON FDDI
1150      NUMBER OF INSTRUCTIONS ; 1
1151      INSTRUCTION NAME ; SUBTRACT 1 FROM TICK.2A
1152    NAME = SEND FDDI 2B FRAME
1153      NUMBER OF INSTRUCTIONS ; 1
1154      INSTRUCTION NAME ; READ G3 DATA FROM BUFFER
1155      NUMBER OF INSTRUCTIONS ; 1
1156      INSTRUCTION NAME ; SEND DATA PACKET ON FDDI
1157      NUMBER OF INSTRUCTIONS ; 1
1158      INSTRUCTION NAME ; SUBTRACT 1 FROM TICK.2B
1159    NAME = RECEIVE AUDIO FRAME
1160      NUMBER OF INSTRUCTIONS ; 1
```

```
1161      INSTRUCTION NAME ; WRITE ASYNC DATA TO BUFFER
1162      NUMBER OF INSTRUCTIONS ; 1
1163      INSTRUCTION NAME ; SEND DATA PACKET TO CADU GEN
1164    NAME = RECEIVE G2 DATA FRAME
1165      NUMBER OF INSTRUCTIONS ; 1
1166      INSTRUCTION NAME ; SEND G2 DATA TO RS CODE GEN
1167      NUMBER OF INSTRUCTIONS ; 1
1168      INSTRUCTION NAME ; WRITE ASYNC DATA TO BUFFER
1169      NUMBER OF INSTRUCTIONS ; 1
1170      INSTRUCTION NAME ; SEND DATA PACKET TO CADU GEN
1171    NAME = RECEIVE G3 DATA FRAME
1172      NUMBER OF INSTRUCTIONS ; 1
1173      INSTRUCTION NAME ; SEND G3 DATA TO CRC CODE GEN
1174      NUMBER OF INSTRUCTIONS ; 1
1175      INSTRUCTION NAME ; WRITE ASYNC DATA TO BUFFER
1176      NUMBER OF INSTRUCTIONS ; 1
1177      INSTRUCTION NAME ; SEND DATA PACKET TO CADU GEN
```

VALIDATION OF PRIORITY FDDI LAN SIMULATORS

John Calvin Watson
Stephen Horan

Dept. of Electrical and Computer Engineering
New Mexico State University
Box 30001, Dept. 3-O
Las Cruces, New Mexico 88003

ABSTRACT

This paper describes GPSS/H, Network II.5, and Block Oriented Network Simulator (BONeS) models of a ten-node, priority-sensitive FDDI LAN. The performance of the rings are compared with theoretical results for these configurations. The purpose for creating the FDDI ring simulation models is to verify model performance of and identify differences in the simulation packages. These differences become important when simulating special features of the FDDI MAC protocol such as the effect of packet collisions on the ring and clobbered overhead bits within a transmitted packet. As a result of these simulations, considerable differences are found in the degree of ease in configuring a simulation model, the necessary run times, and the level of accuracy of the results.

INTRODUCTION

The Fiber Distributed Data Interface (FDDI) was developed using fiber optics and a token ring media access control (MAC) protocol to provide a 100-Megabit-per-second data transfer rate between high performance peripherals. Priority levels can be added to the algorithm for sending the data across the LAN to manage the traffic flow when differing types of service are needed. Several articles have been written describing the theory and performance of the FDDI token ring MAC protocol which enables the high data transfer rates for both synchronous and asynchronous data without violating the minimum guaranteed synchronous bandwidth at each station. The purpose of this paper is to apply the theories developed for the FDDI token ring performance to

benchmark the ease-of-use and performance of three commercial simulation software packages in simulating priority-sensitive FDDI Local Area Networks. Simulation results obtained using GPSS/H [1], Network II.5 [2], and BONeS [3] FDDI token ring simulation models will be presented and compared with the theoretical results. After validation of the simulators and benchmarking the FDDI LAN models, the packages will be used to simulate various FDDI LAN configurations such as the proposed Space Station Freedom onboard data management system [4] shown in Figure 1 where FDDI LAN's will provide the backbone for transferring payload and core engineering data.

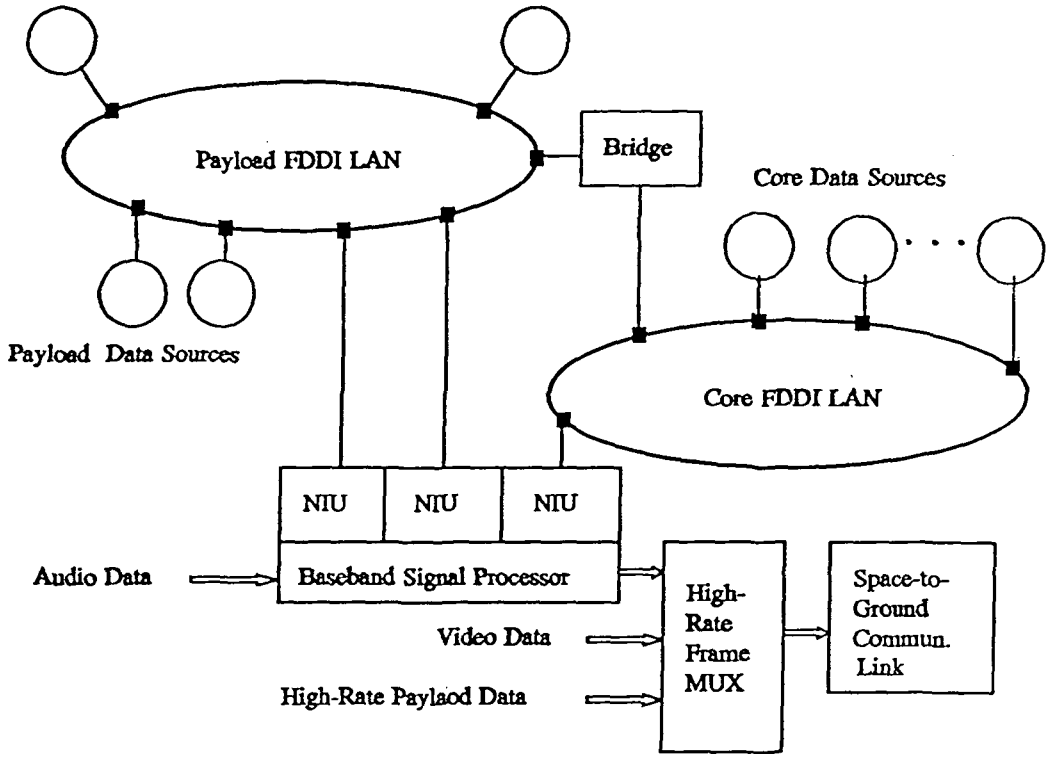


Figure 1 - SSF Onboard FDDI LAN Structure.

FDDI TOKEN RING MAC PROTOCOL

The FDDI token ring media access control protocol was formalized by the American National

Standards Institute [5] and is intended to interconnect high-performance engineering workstations and computer mainframes using fiber optic cable. The FDDI token ring MAC protocol provides each user with a minimum guaranteed synchronous bandwidth which will be satisfied before asynchronous data transmission will be performed on the ring. This ensures that time-critical synchronous data such as digitized voice and real-time control will not be distorted. The FDDI token ring MAC protocol uses a token to control ring access and station token holding time. The speed at which the token completes its journey around the FDDI ring determines the transmission bandwidth of each station. During initialization, each station computes its target token rotation time (TTRT) which is the maximum time allowed for the token to circulate the ring and return to the sender without violating that station's bandwidth requirements. The minimum token rotation time required to satisfy the synchronous bandwidth requirements of all active stations on the ring will be selected during the FDDI ring initialization process. The FDDI token ring MAC protocol enables the FDDI LAN to achieve high data transfer rates by allowing each station the authority to regenerate the expired token without waiting for residual frame data bits to be received which reduces the effect of ring propagation delays.

Eight priority levels are available in the FDDI token ring MAC protocol for the transmission of asynchronous data. Each priority level specifies the time duration (bandwidth) in which the station receiving the token may transmit asynchronous data. Asynchronous data transmission by a station can be performed only if the target token rotation time has not expired. If the TTRT has expired, the station's Late_Ct is incremented. The Late_Ct indicates the number of times the token rotation timer exceeded the target token rotation timer at a station. Should a station's token rotation timer exceed the TTRT with the Late_Ct not zero, ring recovery will be initiated.

A station determines whether the TTRT has expired by comparing the TTRT with the station's Token Rotation Timer (TRT) which measures the time interval between token reception by the station. If the TRT exceeds the TTRT, asynchronous data transmission cannot be performed. The TTRT affects only the asynchronous data transmission of a station since the FDDI token ring MAC protocol guarantees that a station's synchronous bandwidth will be preserved. M.J. Johnson [6] has formally proven that the maximum token rotation time of the FDDI token ring MAC protocol does not exceed twice the target token rotation time (TTRT) regardless of the offered load.

FDDI TOKEN RING UTILIZATION

The FDDI Token Ring MAC protocol was designed to accommodate varying data traffic load levels. The FDDI token ring will always sustain the minimum synchronous offered load regardless of the level of asynchronous data traffic intensity. Under heavy offered traffic load conditions, the FDDI token ring may be restricted to accommodate only synchronous and high priority asynchronous data packets. As the offered load level drops, asynchronous data packets of lower priority may be transmitted. The target token rotation time for each asynchronous data traffic priority class determines which asynchronous data packets are transmitted. The FDDI token ring MAC protocol was designed to keep the FDDI token ring fully utilized.

The token rotation time will depend on the token hold times for each station on the FDDI ring and the station to station propagation delay. The medium propagation delay for the fiber optic cable specified in the FDDI standards is approximately 5 microseconds per kilometer [2]. D.

Dykeman and W. Bux [7] analyzed the FDDI token ring MAC protocol and developed the following equation for maximum throughput of the FDDI ring:

$$\gamma_{MAX} = \frac{(n * tot\_tx\_time + n^2 * tx\_window)*\upsilon}{n * tot\_tx\_time + n^2 * tx\_window + (n^2 + 2n + 1)*r\_I} \quad (1)$$

where n = *number of active stations*
υ = *transmission rate*
tx_window = $T\_Opr - r\_I$
T_Opr = *token-holding time threshold for the asynchronous priority level in use*
r_I = *total ring latency time (propagation delay plus sum of station latencies)*
tot_tx_time = $CEILING(tx\_window/F)*F$
F = *frame transmission time*

Taking the limit of equation (1) as the number of stations goes to infinity, D. Dykeman and W. Bux developed the following result:

$$\lim_{n\to\infty} \gamma_{MAX} = \frac{T\_Opr - r\_I}{T\_Opr} * \nu \quad (2)$$

The maximum throughput obtained using (1) will then be compared with the results given by the simulation packages to verify the GPSS/H, Network II.5, and BONeS FDDI token ring simulation models.

GPSS/H MODEL

A 10-node FDDI ring simulation model was developed using GPSS/H. GPSS/H is a general, high-level simulation language and therefore does not include a software model for the FDDI ring

or any other communications or computer structure. Therefore, the 10-node FDDI ring model was developed from using the standard protocol definition. The purpose for developing the GPSS/H model was to verify the internal operation of the token rotation timers and logic of the FDDI token ring transfer device protocol by giving direct access to the details of the protocol. The GPSS/H model also provides verification of the Network II.5 and BONeS model simulation results because the latter are, essentially, black boxes whose internals are not under direct control of the user other than through the parameters passed to the user interface. An advantage to using GPSS/H is that the model can be tuned to include as much detail as necessary to describe the FDDI ring. The cost for this tuning access is the complexity of the GPSS/H program which will increase as you improve the detail of the FDDI ring simulation model. The software development of the GPSS/H model was extensive compared to the Network II.5 and BONeS models which include internal software modules to simulate the FDDI transfer device protocol. The GPSS/H model can usually perform a 50-second simulation of the 10-node FDDI ring model in less than five minutes depending on the system load. The relatively rapid execution time for the GPSS/H model enables FDDI ring performance plots to be obtained with minimal delay. An advantage to using discrete-event simulation modeling such as GPSS/H is that the simulation duration can be increased without increasing the number of transactions since transactions are destroyed during the course of the simulation.

Each station in the GPSS/H FDDI token ring local area network was modeled as shown in Figure 2. In the GPSS/H model, each transaction represents the received token. The Generate block shown in Figure 2 generates the token. The Gate block allows the token to be received only if the token is inactive at the other nodes. The Gate block also permits the data station to

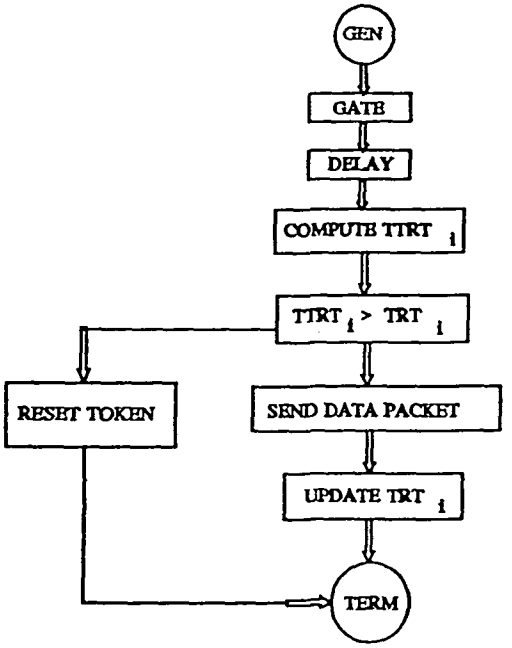


Figure 2 - GPSS/H Data Node Model.

receive the token only if it is the station's turn to send data on the ring. In a hardware implementation of the FDDI ring the scheduling of stations would be performed by their physical location on the ring. However, the GPSS/H software implementation of the FDDI ring must ensure that all stations receive equal access to the ring. The Delay block models the ring propagation delay between nodes on the FDDI ring. The data station computes the Token Rotation Time (TRT) by subtracting the previous value of the simulation clock time when the token was last received from the current simulation clock time. The data station compares the TRT with the Target Token Rotation Time (TTRT) for the asynchronous data priority level assigned to the data. If the TRT is less than the asynchronous TTRT, there is residual token holding time for the node to send asynchronous data packets on the FDDI ring. Asynchronous data packets will be sent on the FDDI ring until the TTRT expires or all of the queued data

packets have been sent. The number of queued data packets depends on the station offered load which can be varied for making throughput performance plots. If the TTRT is less than the TRT, insufficient time remains for asynchronous data transmission at the node and the token must be reset.

NETWORK II.5 MODEL

The CACI Network II.5 simulation software includes the FDDI token ring bus configuration as one of its transfer device options. The Priority Token Ring protocol modeled by CACI allows user selection of the Token Passing Time, Synchronous Allocation, Minimum Synchronous Priority, and Target Token Rotation Times for each station. The Token Passing Time represents the time required to pass the token between stations. The Network II.5 package does not explicitly model the FDDI data frame, rather, for each block of data transmitted, an overhead time for framing bits can be specified to emulate the frame overhead characteristics. The Network II.5 FDDI ring model assumes that stations are equally spaced around the FDDI ring. The Synchronous Allocation represents the guaranteed amount of time that a station may hold the token. The Synchronous Allocation ensures that all stations will be allotted the minimum synchronous bandwidth as specified in the FDDI Token Ring Media Access Control Protocol [5]. The Minimum Synchronous Priority restricts usage of the station minimum synchronous bandwidth to the desired transmission priority level. The Target Token Rotation Times specify different asynchronous transmission priority levels at each station.

The Network II.5 simulation model for the FDDI local area network was developed using ten data stations where each data station was modeled as shown in Figure 3. The Data Packet

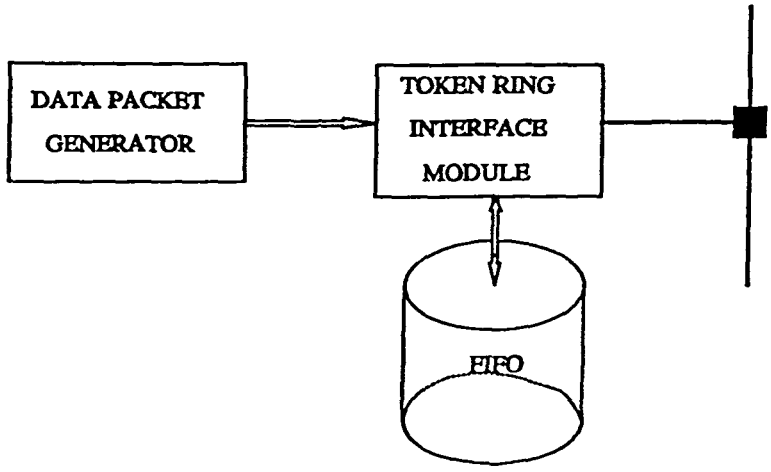


Figure 3 - NETWORK II.5 Data Node Model.

Generator in Figure 3 represents the processing element which generates data packets to be sent on the FDDI ring. The packet arrival rate is adjusted by changing the packet interarrival time setting. The packet generator processing element also sets a semaphore flag to indicate the number of packets generated. The packet accounting information provided by the semaphore is required to control read/write access to the first-in-first-out (FIFO) buffer. The FIFO buffer was included to determine the storage capacity required to accommodate the data packet queue at each of the Token Ring Interface Modules. The Token Ring Interface Module for each station transfers data packets from the FIFO buffer to the FDDI token ring when it receives the token. The FIFO buffer was included to accommodate data packets which may require temporary storage while waiting for access to the FDDI token ring. Since the FDDI token ring MAC protocol guarantees the allotted synchronous bandwidth to each station, the FIFO buffer will only

store asynchronous data packets.

BLOCK ORIENTED NETWORK SIMULATION (BONeS) MODEL

The Block Oriented Network Simulator (BONeS) includes a detailed simulation model of the FDDI media access control (MAC) protocol. The FDDI MAC protocol is modeled using discrete network devices such as traffic generators, data insertion modules, delay modules, memory units, arithmetic units, and timers. The BONeS version of the FDDI MAC protocol simulates the MAC_Data.Req, MAC_Data_Status.Ind, and MAC_Data.Ind service primitives specified in the FDDI standard. The BONeS software package includes the FDDI MAC node in its database which can be copied into a simulation model of the FDDI ring. The flexibility of the BONeS FDDI simulation model results in part from the use of data structures to store the data fields of the token and frame. For example, the frame data structure includes fields for the Source ID, Destination ID, Service Class, Priority, Information, Frame Length, Time Received by MAC, and Time Exited MAC Queue. The number of data fields can be increased as required to include more detail in the FDDI simulation model. Similarly, the token data structure includes data fields to define the Source which created the token, the Source which captured the token, token Length, Time Received, and Total Correction Time.

The BONeS version of the FDDI MAC protocol can be applied to the simulation of data packets which have been corrupted on the ring by altering the traffic generator parameters within the FDDI node module. The BONeS version can also be used to simulate data stations which change the Priority levels of each frame. The Priority Level of an FDDI node can be modeled using a variable which allows it to change after each frame transmission. The BONeS version

of the FDDI MAC protocol is very detailed and complete. Consequently, simulation time using BONeS is excessive (benchmark simulations used here generally required 36 hours or more user time on a SUN-4). The BONeS 10-node FDDI ring simulation model shown in Figure 4 was developed using the supplied FDDI MAC node simulation model included in the BONeS FDDI database. The FDDI node parameters were altered to reflect the same model as described in the GPSS/H and Network II.5 simulation models. The BONeS simulation package allows module parameters to be iterated if necessary for the purpose of making data plots. The BONeS 10-node FDDI ring simulation model shown in Figure 4 was developed for the simulation benchmarks studied here.

SINGLE ASYNCHRONOUS PRIORITY CLASS SIMULATION RESULTS

The GPSS/H, BONeS, and Network II.5 token ring model simulation results were compared with the theoretical results for the Single Asynchronous Priority Level Case described by D. Dykeman and W. Bux as given in equation (1) above. A single asynchronous priority level was assigned to each of the ten stations connected to the FDDI ring in the GPSS/H and Network II.5 simulation models. The data traffic generators were set to deliver a 100 Mbps offered load to the FDDI ring. Maximum throughput was determined by summing the total number of completed data packet transfers on the FDDI ring, multiplying this sum by the packet length, then dividing by the simulation duration. The maximum throughput as a function of ring latency plot shown in Figure 5 describes the GPSS/H, BONeS, and Network II.5 simulation results for ten stations, a 12620-bit packet length, and a single asynchronous priority level for each data station with TTRTs of 5 milliseconds and 10 milliseconds. The ring latency was varied from 0.110 to

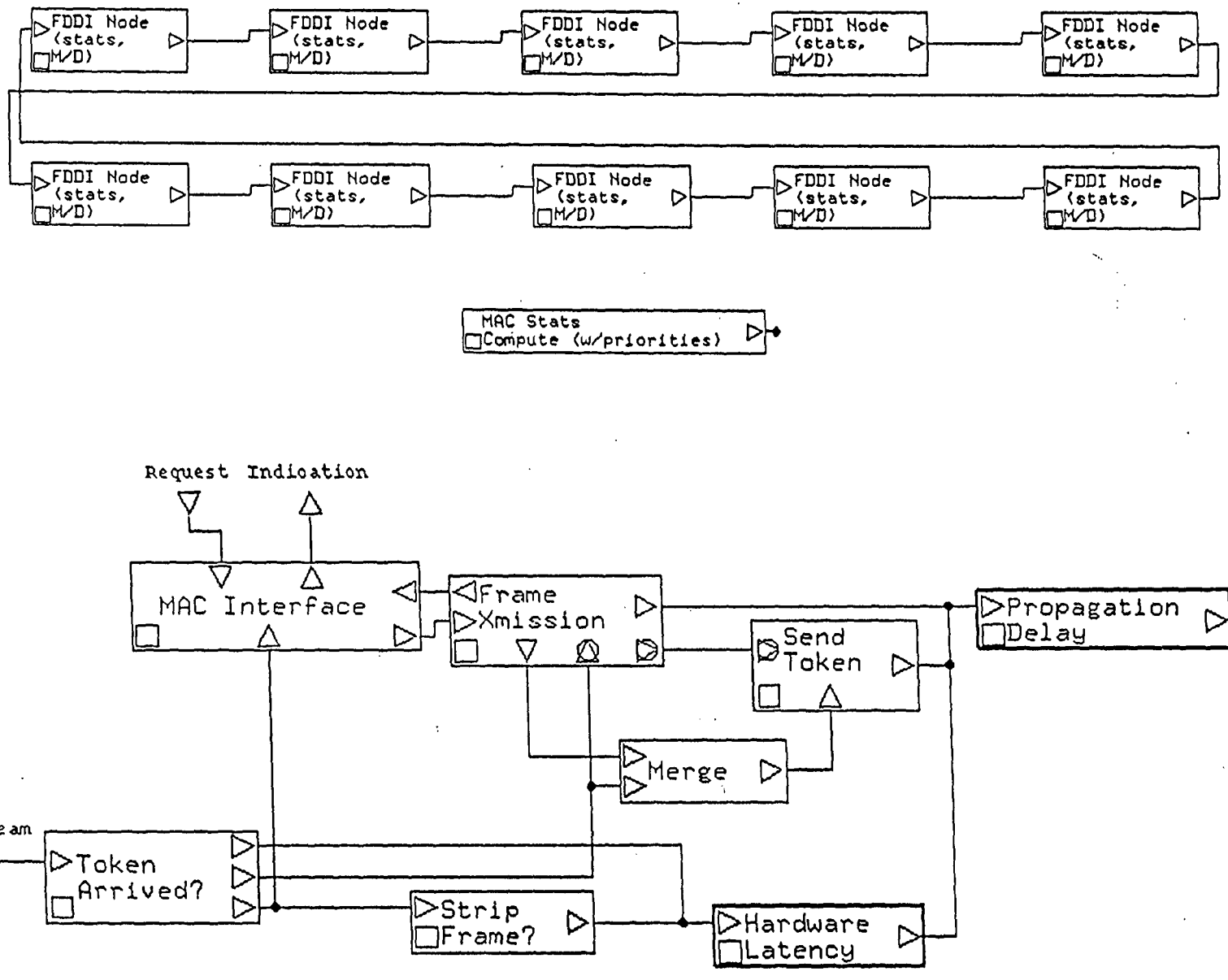
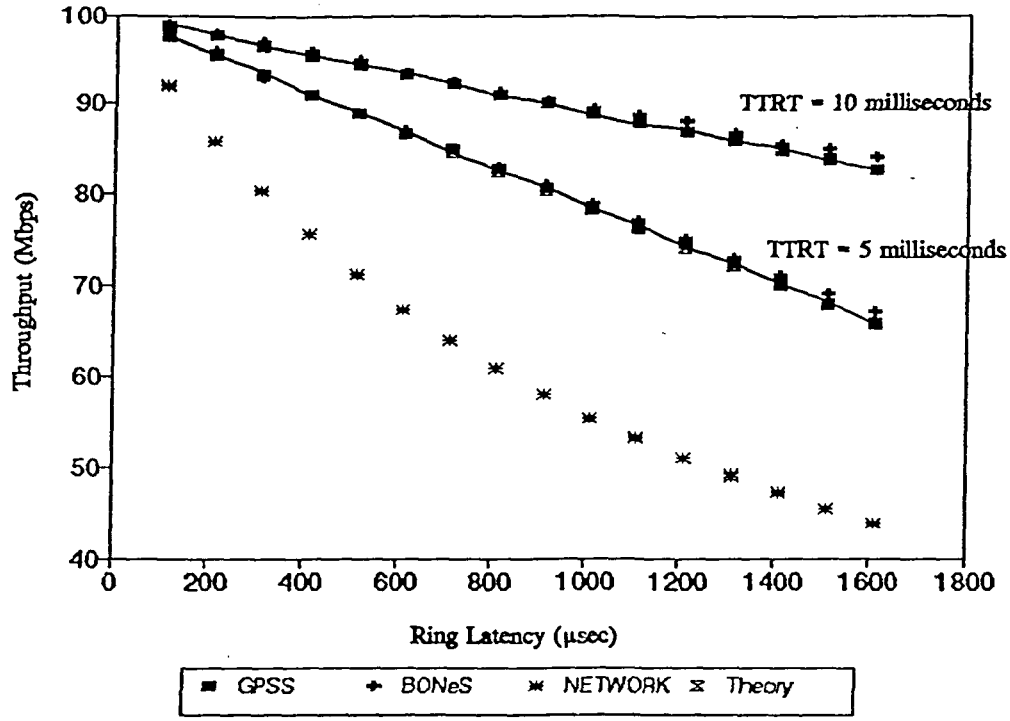


Figure 4. BONeS 10-node FDDI Ring Simulation Model



1.610 μsec which represents FDDI rings from 22 kilometers to 322 kilometers in length. The GPSS/H model simulation results deviated from the theoretical approximation for maximum throughput developed by D. Dykeman and W. Bux by less than one percent. The BONeS results also agree with the theoretical results very well. The Network II.5 models did not follow the predicted results very well at all. No amount of varying the configuration of the model changed the Network II.5 performance in this simulation experiment. The theoretical maximum throughput vs ring latency for ten stations determined from (1) was included in the plot of Figure 5 to verify model simulation results. The theoretical maximum throughput results as the number of stations goes to infinity do not differ significantly from the theoretical ten station results. The Maximum Throughput vs Ring Latency plot in Figure 5 shows that the maximum throughput of the FDDI ring will decrease as you increase the size of the ring. The size of the

FDDI ring will have a greater impact on utilization than will the number of stations if the station latency is not excessive compared to the station to station propagation delay.

The target token rotation time (TTRT) limits the residual time remaining on the token for asynchronous data transmission. The greater the value of the TTRT for a given asynchronous data priority class, the greater the residual time remaining on the token for asynchronous data transmission on the FDDI ring. Asynchronous data transmission will commence only if the TTRT for the asynchronous data priority level of the frame to be transmitted equals or exceeds the station TRT. When one increases the TTRT for an asynchronous data priority class, the maximum throughput increases as shown in the Maximum Throughput vs TTRT plot shown in Figure 6. The results shown in Figure 6 were from the simulation of ten identical stations where

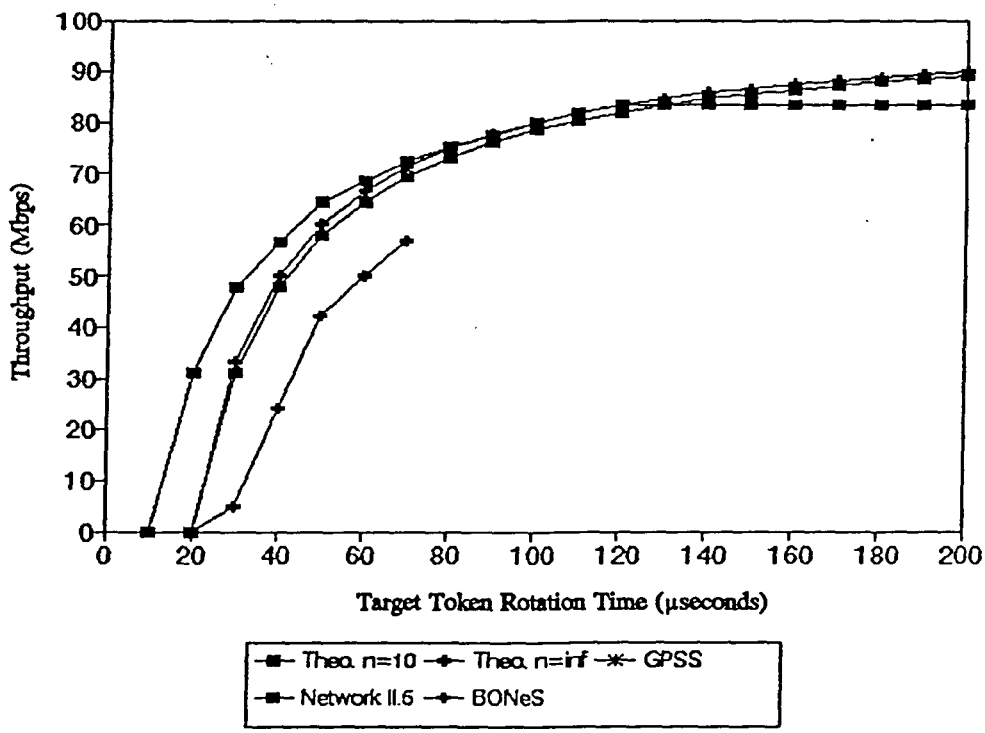


Figure 5 - Ring Throughput vs. TTRT.

each station has the same TTRT for asynchronous data transmission. The TTRT was varied from 0 to 200 μsec with a 1000-bit packet length. The FDDI ring latency was 20 μsec which represents a ring length of about 4 kilometers. The GPSS/H results for Throughput vs TTRT agree with the theoretical results from Dykeman and Bux [6] within one percent. The Network II.5 model results for Maximum Throughput vs TTRT deviate from the theoretical results shown in Figure 6 for TTRTs less than 60 microseconds. The Network II.5 model maximum throughput for TTRTs between 60 and 120 microseconds tends to follow the theoretical results shown in Figure 6 until it saturates at about 83 Mbps. The Network II.5 simulation software includes the Priority Token Ring Protocol transfer device which was used to model the 10-node FDDI ring. The code contained within the Network II.5 Priority Token Ring Protocol software module is not available for inspection or alteration by the user. The BONeS 10-node FDDI simulation results tend to follow the theoretical results shown in Figure 6 with a TTRT offset. The BONeS simulation model maximum throughput results shown in Figure 6 required more than 60 hours user time to simulate on a SUN-4. The deviation between the theoretical and BONeS model maximum throughput vs TTRT results may be due to the high level of detail in the BONeS model.

MULTIPLE ASYNCHRONOUS PRIORITY LEVELS

The FDDI MAC protocol allows stations to be assigned priority levels for asynchronous data transmission. Each priority level will have a different target token rotation time assignment. The higher priority stations will have longer TTRT's which will enable them to capture the token for a longer period of time for transmitting asynchronous data. Dykeman and Bux developed a

simulation model for multiple asynchronous priority levels which was used as a reference for the GPSS/H, Network II.5, and BONeS 10-node FDDI ring simulation models. The data stations were assigned the priority levels and asynchronous TTRTs given in Table 1. To emulate the conditions under which Dykeman and Bux developed their throughput estimation, the packet length was 12620 bits and the ring latency was 1.0236 msec in the simulation models. The theoretical results for the multiple asynchronous priority level case described by Dykeman and Bux are shown in Figure 7. The theoretical case describes 11 stations connected to the FDDI ring with three idle stations.

The GPSS/H, Network II.5, and BONeS 10-node FDDI ring simulation models duplicate the theoretical model except for not including the eleventh "inactive" station and reducing the data traffic throughput from stations nine and ten by assigning Priority Level 0. The Throughput vs Arrival Rate Per Priority plot shown in Figure 8 was produced from the BONeS 10-node FDDI ring simulation model. As can be seen in Figure 8, as the offered load increases the higher priority asynchronous traffic occupies more of the FDDI ring bandwidth. Throughput from the lower priority traffic stations falls off as the higher priority stations dominate the FDDI ring allocation as the offered load increases. At an offered load of 1.0, the Priority 7 and Priority 6 data stations have throughputs of about 70 and 30 Mbps, respectively.

Station	Priority	TTRT
1	0	1.5 msec
2	1	6.2 "
3	2	14.0 "
4	3	25.0 "
5	4	39.0 "
6	5	56.2 "
7	6	76.9 "
8	7	100.0 "
9	inactive	
10	inactive	
11	inactive	

Table 1. Priority and TTRT Assignments for Theoretical Model

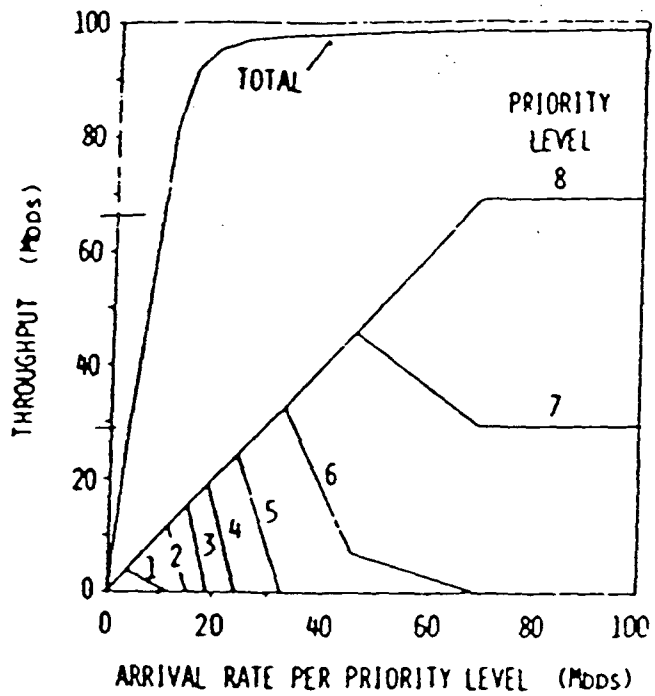


Figure 7. Theoretical Throughput vs Arrival Rate per Priority Level [8]

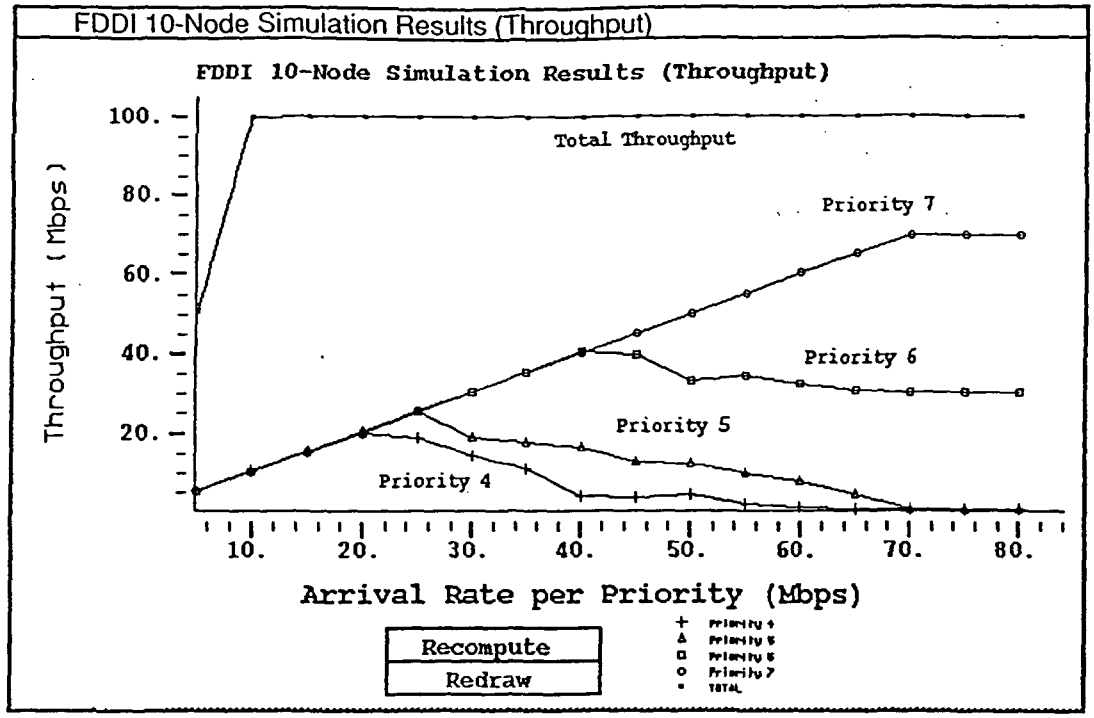


Figure 8. Throughput vs Arrival Rate/Priority for the BONeS Model

A similar result can be observed using the GPSS/H model of the 10-node FDDI ring where the Throughput vs Arrival Rate Per Priority plot is shown in Figure 9.

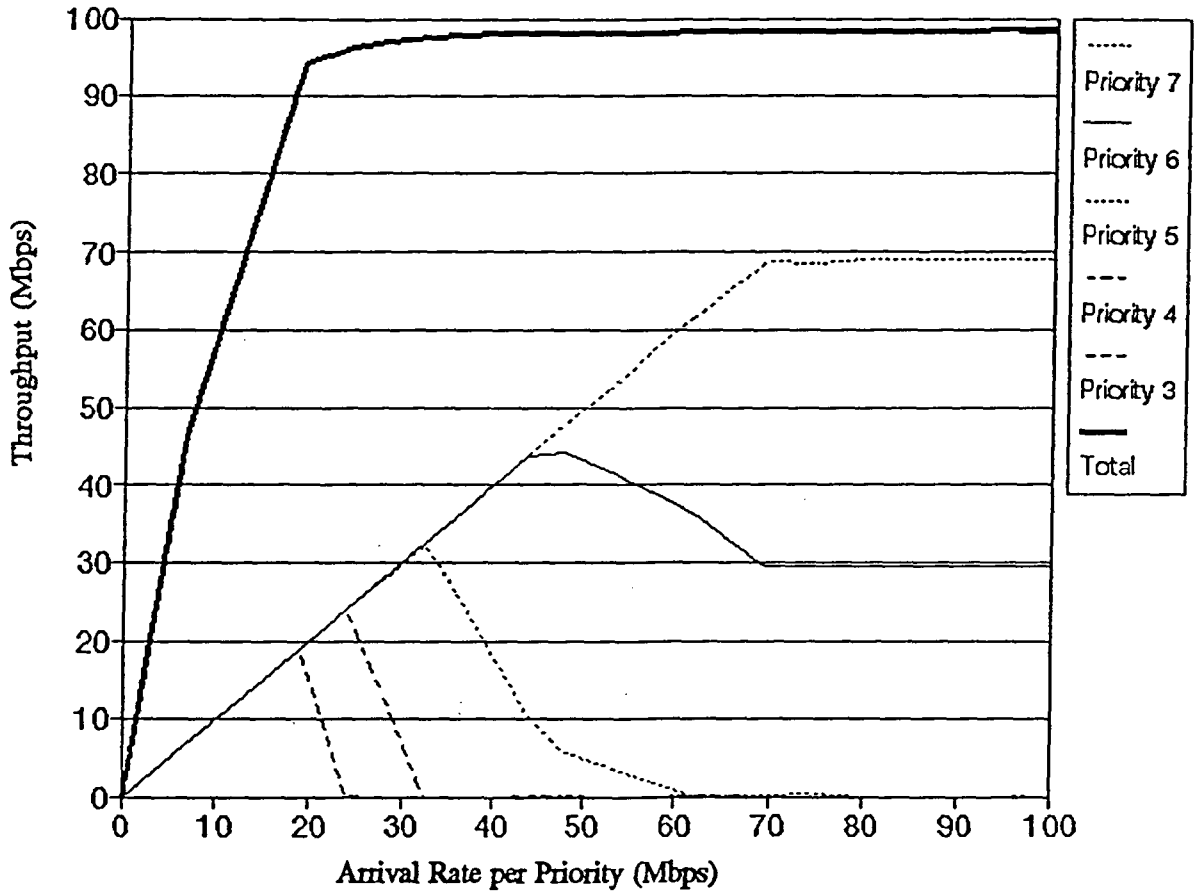


Figure 9. Throughput vs Arrival Rate Per Priority for GPSS/H Model

The Network II.5 simulation model of the 10-node FDDI ring produced an output that showed a constant throughput for all offered loads regardless of the priority level of the data. This leads

to a question in our minds about the level of detail modeled in the Network II.5 FDDI protocol model. Since the user does not have access to the details of the simulation model, it is difficult to know what exactly is happening in this case.

SUMMARY

The GPSS, BONeS, and Network approaches to modeling FDDI networks contain many tradeoffs for the simulation user. Table 2 lists some of those tradeoffs and summarizes our opinions on how the three packages fair in these categories.

CONCLUSIONS

The level of complexity of the FDDI LAN simulation model depends on the problem to be solved. For example, if the desired simulation result is an estimate for the throughput of the FDDI ring when representative stations are modeled, a GPSS/H simulation model which requires little simulation time may be suitable. If the desired result involves analyzing the performance the FDDI ring after the overhead bits in circulating packets have been corrupted, a detailed model such as the Block Oriented Network Simulator would be required. The complex FDDI ring simulation models which include details about the frame and token will generally require many hours of simulation time. A suggested modeling approach would be to use a fast simulation model such as GPSS/H and Network II.5 to develop an approximate solution. Then refine the solution by including specific details about the model in a complex simulation package such as the Block Oriented Network Simulator.

Table 2. Comparison of Experience with FDDI Simulation Models.			
Attribute	GPSS	BONeS	Network II.5
Model Development	extensive hand coding	extensive combination of model primitives	fixed input parameters from a fixed menu of options
Automatic Model Iterations	not available	input parameter	not available
Level of Parameter Detail	as extensive as user codes	extensive	moderate
Ease of Modification	base code needs to be edited	graphical block diagram based user interface with user input of parameters	structured user interface with menus
Relative Execution Speed	fast	slow	moderate
Execution Platforms	mainframe	SUN-4	SUN-4, VAXstation, PC
Single Asynchronous Priority Simulations	excellent agreement with theory	excellent agreement with theory	disagreement with theory
Multiple Asynchronous Priority Simulations	excellent agreement with theory	excellent agreement with theory	severe disagreement with theory

REFERENCES

1. GPSS/H Simulation Software, Wolverine Software Corporation, 1989.

2. CACI Network II.5 Release 6.0 Simulation Software, CACI Products Company, 1990.

3. Block Oriented Network Simulator (BONeS), Comdisco Systems, Inc., 1990.

4. J.E. Smith, D. Willett, and S. Paul, "Overview of the Space Station Communications Networks," IEEE Network Magazine, Sept. 1990.

5. "FDDI token ring media access control (MAC)," American National Standard, X3.139-1987.

6. M.J. Johnson, "Proof that Timing Requirements of the FDDI Token Ring Protocol are Satisfied," IEEE Transactions on Communications, Vol. Comm-35, No. 6, June 1987.

7. D. Dykeman and W. Bux, "Analysis and Tuning of the FDDI Media Access Control Protocol," IEEE Journal on Selected Areas in Communications, Vol. 6, No. 6, July 1988.

2. PARALLEL ARCHITECTURE PERFORMANCE STUDIES

Javin M. Taylor
John Polson
A. K. Petersen

2.1 INTRODUCTION

Section 2.2 describes discusses various hardware platforms for the Data Handling Service. The previous semi-annual reports [10-12] and references [3] and [4] have discussed in detail the DHC homoforms associated with an architecture presently under development in the Electrical & Computer Engineering Department at New Mexico State University. This architecture, conceived by Dr. Eric Johnson, is based on a 64-processor global memory message passing architecture comprised of 25 MHz microprocessors. Continuing analysis shows that this architecture can accommodate four concurrent 300 Mbps data streams resulting in an aggregate bandwidth of 1.2 Gbps. This architecture is an excellent choice of the Data Handling Center. The inclusion of a DMA bus has increased the efficiency of this architecture.

In addition, different implementations are discussed and compared, issues requiring additional research are mentioned, and recommendations are made.

Section 2.3 describes construction of a NETWORK II.5 simulation model of the Data Interface Facility (DIF). The model is concerned with data buffered and sent out by the DIF, and in particular, sent to the Data Handling Service. In an attempt to overcome several problems in the use of the NETWORK II.5 simulator, several other simulators were evaluated. GPSS/H was chosen and a model of the DIF was written to parallel the NETWORK II.5 version.

Section 2.4 identifies a commercially available storage facility, called the Mass Storage System (MSS) which can handle the extraordinary storage required to archive the data flowing out of the DHC.

2.2 HARDWARE PLATFORMS FOR THE DATA HANDLING SERVICE (John Polson)

2.2.1 Introduction

The Electrical and Computer Engineering Department at New Mexico State University in Las Cruces, New Mexico has been studying advanced space data systems for NASA under the Advanced Telemetry Processing Pilot Program since 1988. The ATP^3 research project has been studying the space-based network proposed by NASA. This section of the final report is a summary of the work done by myself and others in relation to the Data Handling Center.

In the Space Station era, the volume of telemetry data is enormous. In 1988, projected bandwidths were in excess of 900 million bits per second (Mbps). In 1988, the available

processing power was not capable of handling this data rate. Analysis and simulations indicate this is no longer the case; there are hardware platforms capable of performing level zero processing on high speed telemetry data.

In section 2.2.2., a summary of areas investigated is presented with important details and revelations. These areas include preliminary investigations of multiprocessor systems, detailed simulation studies of the Virtual Port Memory Multiprocessor Architecture, and analytical studies of several alternatives.

In section 2.2.3., different implementations are discussed and compared. In section 2.2.4., issues in this part of the report requiring additional research are discussed. The most important issue considered by this phase of study requiring further research is the cost of designing and implementing the software for the Data Handling Service. Finally, in section 2.2.5., final recommendations are made as a result of this research.

2.2.2 Background

The research done on the Data Handling Service can be divided into three phases. The first phase is the initial investigation about the characteristics of the Data Handling Service (DHS) and is discussed in section 2.2.2.1. The second research phase is a simulation of a system that fits the DHS problem well. The second phase is discussed in section 2.2.2.2. The third phase is an examination of alternatives and is presented in section 2.2.2.3.

2.2.2.1 Preliminary Investigation

Initial problem requirements indicated the DHS is required to handle four 300 Mbps serial data links simultaneously. There are no SISD computers available that can handle one data bit every 3.3 nS on four channels. Thus, alternative approaches are considered.

SIMD computers are not a good choice because the packets arrive serially. Also, the processing for the three grades of service is slightly different with some grades maintaining packet order (scientific data) and others putting in fill data for lost packets (video).

The data is sufficiently diverse to eliminate systolic arrays. Data flow machines are not sufficiently well developed to lend themselves to an easy implementation and easy maintenance. The key point pushing the choice of MIMD for the DHS is the independence of one packet from another. This is particularly true when the packets are on different virtual channels.

Initially, a hypercube architecture was examined as a possible DHS platform. This was quickly dismissed due to data movement problems and required Operating System features. There is simply insufficient bandwidth between nodes and from the external world to the hypercubes to facilitate a Data Handling Service. In short, hypercubes are number crunchers not real time data movers.

This leaves the global memory MIMD computers as the logical alternative. Since most of the level zero processing uses integer instructions, high speed floating point units will be used very little. Pipelined arithmetic units capable of billions of floating point operations per second are useless in the DHS. Any pipelining is done at a much higher level. A pipeline with data packets flowing through it is a much better approach. Finally, problem requirements indicate the DHS handles the data packets in real time. This implies a multitasking operating system. Thus, the second phase of research was started. In the next section, a simulation of a Virtual Port Memory Multiprocessor is presented.

2.2.2.2 High Level Simulation of the Virtual Port Memory Machine

The Data Handling Service is a small piece of a much larger system. The DHS must be reliable and flexible. The Data Interface Facility is the data source for the DHS. The DHS processes this data and passes it along to the end users. The Virtual Port Memory machine can satisfy the requirements of the DHS suggested by Computer Sciences Corporation in [1] and [2]. A more complete discussion of this section may be found in [10] and [11].

2.2.2.2.1 NASA's Spaced Based Network Requirements

All space communications are routed through the TDRSS, Tracking Data Relay Satellite System. There will be two TDRS in the east and two in the west. All four of the 300 Mbps channels are directed to NASA's ground terminal at the White Sands Test Facility near Las Cruces, New Mexico. The ground terminal feeds a facility called the Data Interface Facility, DIF. The processing done at the DIF has not been finalized. However, the data rate into and out of the DIF is physically limited 1200 Mbps as 4 independent, serial 300 Mbps channels.

The data produced by the DIF is sent to the Data Handling Service. Level zero processing is done at the DHS before the data is finally relayed over land based networks or domestic satellite networks to the owners of the data. Level zero processing involves reordering of packets, conversion of some data from encoded form to engineering units, reversing the order of priority playback data, and archiving data for up to two years [3,4]. Figure 1 shows how the space based network components are connected.

The three major data components are video, audio, and experimental data. The video data is high priority and high rate. Audio data is low rate and high priority. These two data types can tolerate some errors. The final data type is experiment data. This data can tolerate very few errors but does not have to get to the user immediately, only correctly. This errorless requirement may mean longer download times.

Telemetry is transmitted in frames called Virtual Channel Access Protocol Data Units or VCA-PDUs. These frames have headers and trailers that contain virtual channel information and Reed-Solomon check symbols. The VCA-PDUs are fixed length to ease synchronization. A complete description of VCA-PDUs can be found in [3,4].

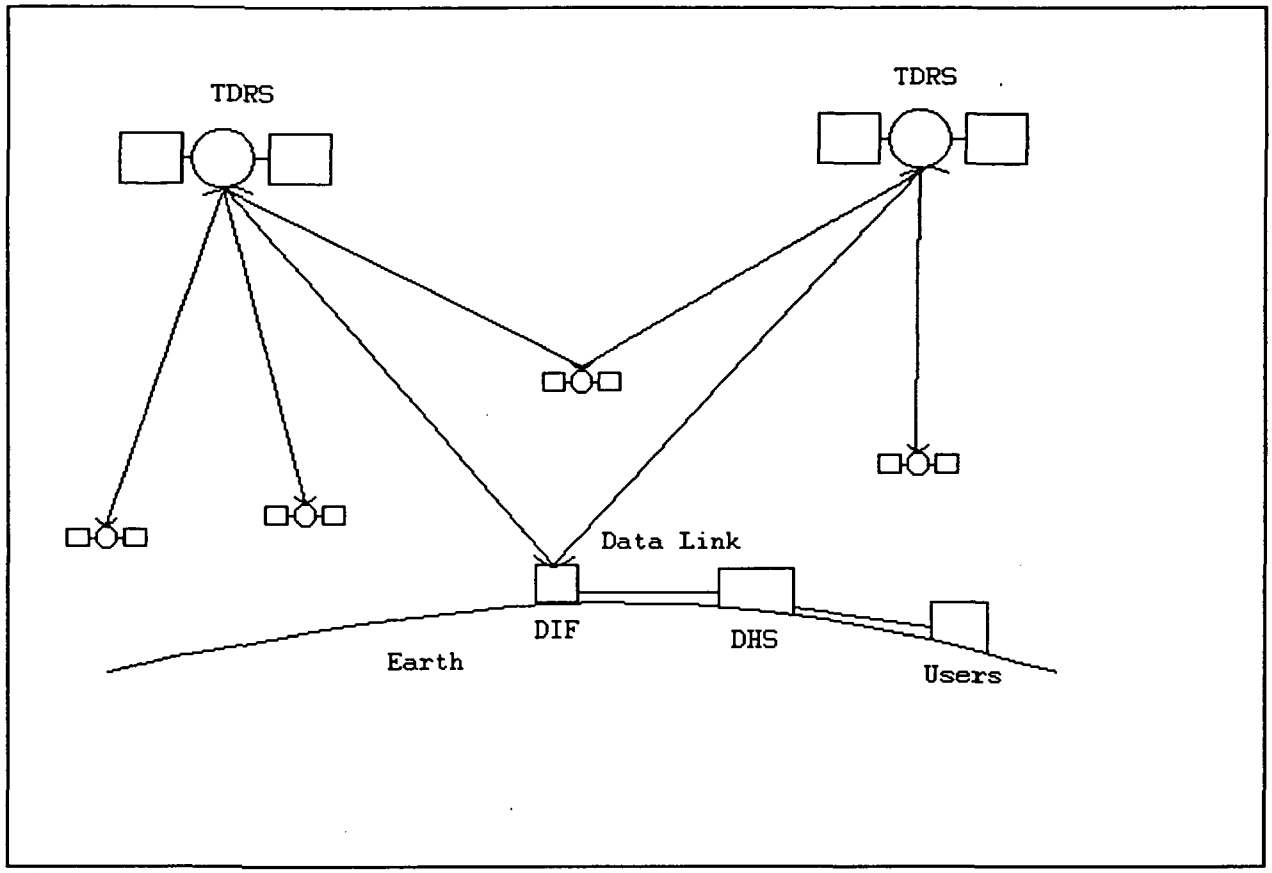


Figure 1 - Space Based Network

The DHS performs two major procedures on a VCA-PDU. Packet processing is performed and the information is finally passed on to the end users over land based or domestic satellite networks [1]. Computer Science Corporation specifies the transformations to be performed by the DHS in a two part report [1,2].

The packet processing is broken up into 5 major steps. Sequentially, the steps are accept input data, demultiplex by virtual channel number, demultiplex by spacecraft identification number and application identification number, reassemble packets, and select storage method. The packet reassembly is monitored and statistics are kept for the DHS operators [1]. The accept input data step is simply taking the data from the transmission media or physical layer and buffer it in memory for processing. The rate at which CADUs arrive is enormous. A single processor can not handle all of the data. Thus, the synchronization markers are removed and the VCA-PDUs are divided among many processors by virtual channel number and spacecraft id number. This reduces the rate at which VCA-PDUs must be processed by a single processor. The VCA-PDUs are the product of the Data Link Layer encapsulation service. They are smaller pieces of large blocks of information, or they are small blocks of information with some filler. The pieces of large blocks must be ordered and assembled. This is done by the reassemble packet function. Finally, some of the data must be archived before being passed onto the end users. This is done in the select storage phase.

There are two steps for distributing data. They are retrieve data from the database and distribute data over the networks. This procedure is also monitored and statistics kept for the operators [1]. This seems simple, however, there is a conversion between network protocols at this step when the information is taken from the space based network and put on a domestic network.

2.2.2.2.2 Virtual Port Memory Multiprocessor Architecture

A Virtual Port Memory Multiprocessor Architecture is a Global Memory Message Passing, GMMP, machine. VPM machines are described by Johnson [5] and are general purpose computing machines. The VPM machine is defined as follows:

> A virtual port memory multiprocessor architecture provides each process of a computation or concurrent system with a private virtual address space and pass by value message passing primitives, based upon an underlying hardware structure consisting of a shared memory, equally accessible to all processors, and a pass by reference message network [5].

The example machine which Johnson describes may have a total of 256 processing elements, I/O controllers, and user interface processors. A four processor prototype of this machine is under development in the Electrical and Computer Engineering Department, New Mexico State University in Las Cruces, New Mexico.

The prototype of the VPM architecture at NMSU is shown in Figure 2. The Interprocessor Message Bus, IMB, is a relatively low bandwidth bus that messages are passed over. Messages are passed between processes. The message unit at the Processing Element (PE), Input Output Controller (IOC), or User Interface Processor (UIP) catches all messages intended for a process that is running on the attached PE, IOC, or UIP.

The global memory in the VPM Prototype is a paged segmentation scheme. A pointer to a segment of memory is a common message. The receiving processes' segment table is updated to show a new segment, and the page frames in the global memory are marked as "copy on write" which effectively gives both the sending process and the receiving process its own copy of the data. If one of the processes writes to one of the common pages a new copy is created for that process and the two pages are not marked as copy on write. Similarly, multiple processes can see the same physical pages of memory as long as the copy on write status is maintained [6].

Each PE board is made up of a message unit, a processor, and a large cache memory [5]. The message unit is responsible for sending and receiving messages over the IMB. The large cache memory is used so that the PE can run as long as possible without using the shared Data Transfer Bus (DTB). The width of the DTB is the same as the length of a line in the cache for ease of implementation. The UIP boards run user's interface programs such as UNIX-like shells. The IOC boards handle all Input and Output devices such as line printers, disk drives, and ethernet connections. There are many types of IOCs that are required.

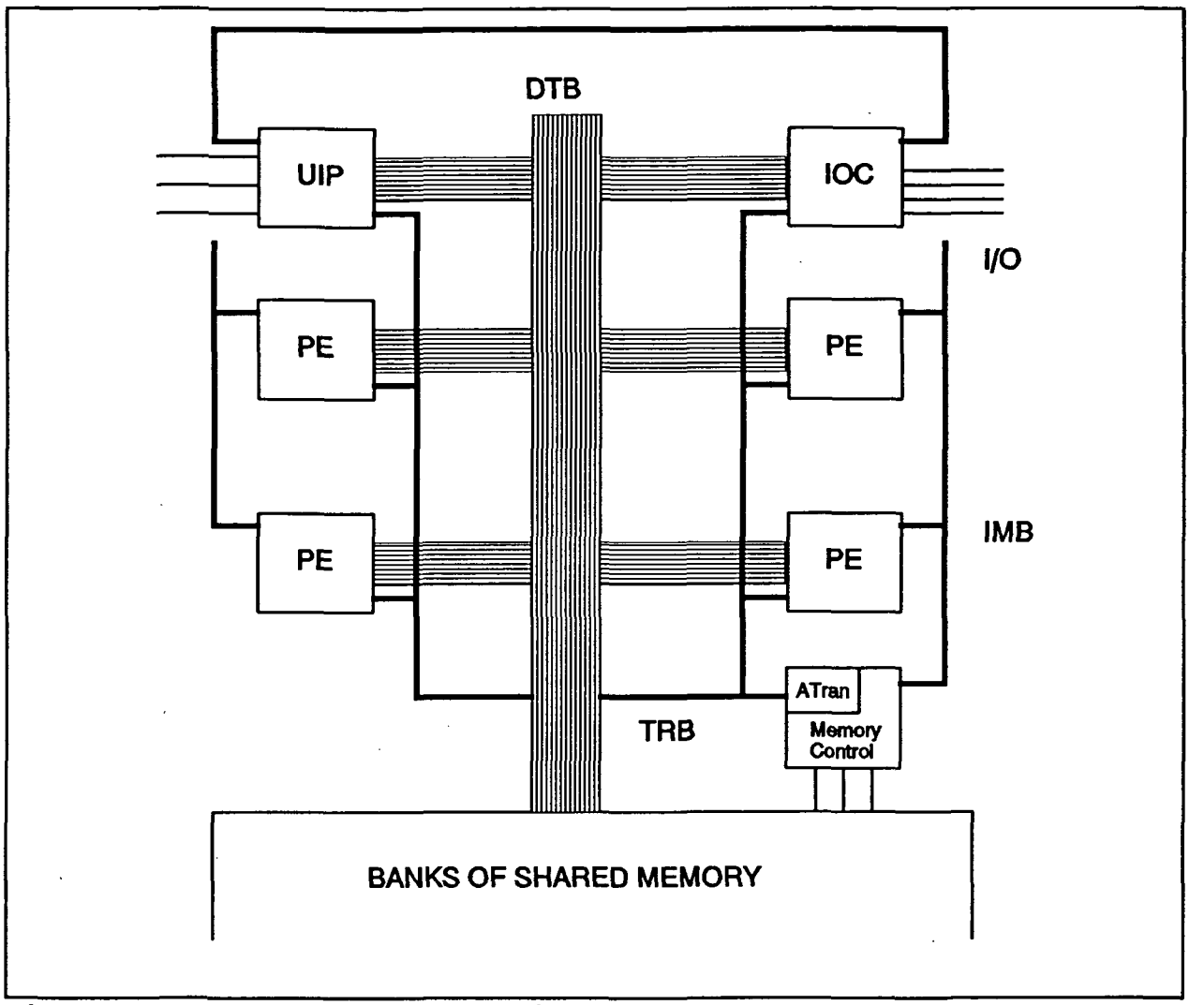


Figure 2 - Prototype VPM Machine

When a processor requests a memory reference that can not be handled by the local cache, a request is made to the global memory for the appropriate location. First, the Transaction Request Bus, TRB, is acquired for the request and the appropriate address is placed on the bus. This reaches the Address Translation cache, ATran, which performs the virtual to physical address translation. This usually takes a single clock cycle to complete. However, if the ATran cache can not handle the translation, a software exception is made to update the ATran cache appropriately and then the translation is completed. Finally, the memory request is queued up in a FIFO register at a memory bank, and the memory cycle is performed. The data is transferred over the DTB.

2.2.2.2.3 Software and Process Communication

There are three pieces of hardware a packet, VCA-PDU, encounters on its way through the Data Handling Service. In order, they are an input IOC, one or more PE, and an output IOC. An input IOC is the piece of hardware that will accept input data, and

demultiplex by spacecraft identification number and virtual channel number. The spacecraft identification number and virtual channel number make up a segmented address space for virtual channels. This segmented address space, VCDU identifier in the VCA-PDU header, is used for the demultiplexing operation. Data frames that consist of fill data are thrown away at the input IOC. Specifically, fill frames are not put into main memory. Once the VCA-PDU is in main memory, a message is sent over the IMB to the process which is responsible for processing that frame. This message contains the segment number where the VCA-PDU is located in memory. Once this message is sent, the IOC releases the segment from its segment table. This effectively gives ownership of the memory that holds the transfer frame to the process that handles the frame. If there is no process responsible for handling the newly arrived frame, a message is sent to an error handling process which checks the header, if possible, and issues appropriate messages to a Data Handling Service error process or to the Data Interface Facility.

The processes that are responsible for handling transfer frames execute on the processing elements in the VPM architecture. There are many identical processors. However, there are not as many processors as there are segmented virtual channels. Thus, many processes may execute on a given processor. Each process handles a specific virtual channel. These processes are different for each virtual channel. However, there are some common functions among processes. Each process will prepare the data for archiving and distribution. However, some data may not be archived, and some data may not be distributed. A few virtual channels will be devoted to priority playback data, and these processes will have to reverse the order of the data bits. The variety of functions performed at this stage is limited only by the number of payloads. When processing is complete, a message is sent to an output IOC. This message, in general, contains a segment that holds a list of segments that have prepared data in them. These segments are given to the IOC as described in [3].

An output IOC is responsible for archiving and distributing the transformed data. The first segment is read into memory. This segment is examined for instructions as to where to send the data, how to archive the data, and where the data is located in memory. The data may be relayed to multiple users over networks and could be archived in multiple locations. However, the data is read from the VPM's main memory just one time. Just like the input IOC, the output IOC does simple operations in the same manner each time.

The global memory in the VPM architecture is used to hold packets as they are processed, to hold all code, and to hold all operating system data structures. This memory requires a very high bandwidth in and an equally high bandwidth out. Thus, the main memory in the VPM machine is made from video random access memory or VRAM [3].

Four input IOCs are required to put the transfer frames into the global memory and pass a message to the process that is handling a particular frame's virtual channel number. The transfer frames arrive by a serial link. This link may be an RF link or it may be a fiber optic link, but this data must be converted to a parallel form for storage into memory.

Also, a 16 bit CRC code must be generated for each packet and stored after it in memory [3].

When a packet has arrived the contents of the VRAM shift register must be moved into the DRAM array. This takes one memory cycle or 80 ns. The 427 ns required to fill up a high speed shift register provides plenty of time for this memory cycle [4].

The interface to the output IOC is not clearly defined. However, the second serial port on the VRAM will be used to removed the data from main memory when necessary. This data will be buffered locally in RAM by the IOC until two functions are complete. These are the archive and distribute data functions described earlier.

It is possible to build a disk drive that has an input and output bandwidth in excess of 300 Mbps. This will be used to buffer the data because a disk drive's memory is non-volatile. Eventually, the data will be moved from this disk drive to some other disk drive. This is very similar to the memory hierarchy used in modern workstations. While the archiving is being performed, the data will be put on the domestic networks destined for the end user.

2.2.2.2.4 Processing Time Models

CACI's NETWORK II.5 simulation package is used to determine the significant parameters for the Virtual Port Memory Multiprocessor Architecture. The width of the DTB, the width of the IMB, the number of PEs, and the average processing time are found. A complete description of the simulation can be found in [3]. A discussion of NETWORK II.5 can be found in [7] and [8].

NETWORK II.5 simulates *processing elements*, *transfer devices*, and *storage devices*. Modules represent processes executing on the processing elements. Processing elements are defined by their instructions. Transfer devices are described by their physical attributes and protocol. Storage devices are described by their access time, resident files, size limitations, and number of access ports. Modules can be triggered or started by sending a message to a processing element.

The simulation model provides the ability to thoroughly analyze the VPM design. There are many design requirements that need to be evaluated. Analysis indicated that the following criteria needed to be evaluated first to determine if the VPM architecture was capable of meeting CSC's functional requirements for the Data Handling Service.

- Width of the Data Transfer Bus
- Width of the Interprocessor Message Bus
- Number of Processing Elements
- Average Processing Time per Packet
- Resident Time in Memory

Simulations are updated to reflect changes in DHS requirements, hardware specifications, and software specifications.

The simulation was used to determine the width of the DTB. The width of the DTB was increased until its simulated utilization was less than 50%. Only powers of 2 were used for the number of bits retrieved and the frequency of operation was fixed at 16 MHz. The low frequency is a requirement because the DTB runs across a backplane bus. It is important to note that on average 1.5 bytes of packet data is accessed for each byte of packet data. This simulates the manipulation of pointers to data and writing them back to the global memory. A DTB running at 16 MHz should be 128 bits wide.

Then the processing time simulation was used to determine the width of the IMB. The messages passed over the IMB were simulated at 128 bits because this is the required length of a message that passes a segment number. An IMB running at 16 MHz should be 16 bits wide.

The simulation was used to determine the number of Processing Elements required to perform a minimum average processing time of 300 microseconds, approximately 1 instruction per byte of data plus message overhead times using a 25 MHz CISC processor. There should be 64 Processing Elements in the DHS.

Finally, the average processing time was increased so that all PEs were near 100% utilized. This average processing time was found to be approximately 425 microseconds. For a 25 MHz CISC processor this is approximately 1625 instructions or 1.6 instructions per byte of data. Considering the quantity of video data and how little processing is done to it, video data is simply passed on to the output IOC by the PE, the average is small. The experiment data will have plenty of time for level zero processing. Also, when the data rate is not running at its peak rate, each packet can take a little longer.

According to the simulation, the maximum time a single packet spent in memory was 1200 microseconds. Each packet requires a whole page frame of memory which is currently 2 Kbytes, and according to Little's law the number of packets in memory does not exceed 176 [9]. Thus, approximately 360 Kbytes of memory is all that is required to buffer packets in memory.

The 128 bit wide DTB implies memory banks that are 128 bits wide. Using VRAM chips, each bank has 4 Mbytes of memory. Thus, 8 banks of memory are used to give the machine 32 Mbytes of global memory. This allows each input IOC 2 banks to use for placing packets in memory. The remaining memory is required for storing code and data structures for the VPM machine.

2.2.2.3 Alternative Approaches

Analysis indicates a system running at 256 MIPS with appropriate I/O bandwidth can handle the DHS. This assumes little interference among processors, and some operating system overhead. The 1200 Mbps requirement need not be met at all times. If it is not met, rate buffering is required. However, a system capable of 256 MIPS will require little rate buffering.

Convex Computer Corporation is a vendor that produces one of the proposed hardware platforms. Convex's C-2 line is capable of handling the DHS function, and some of its short comings will be eliminated in their next generation. Alliant Computer Corporation is another vendor that is currently capable of processing the space-based network packets. The last vendor discussed is Aptec, a maker of I/O busses. A more complete discussion of these alternatives may be found in [12].

2.2.2.3.1 Convex 230i

The Convex 230i computer system is a member of the second generation of Convex Computer Corporation products. The Convex C-2 line is air cooled which greatly reduces the operation costs and maintenance staff. Figure 3 shows a C-240 system. A Convex C-240 system has a mean time between failure of approximately 41 days.

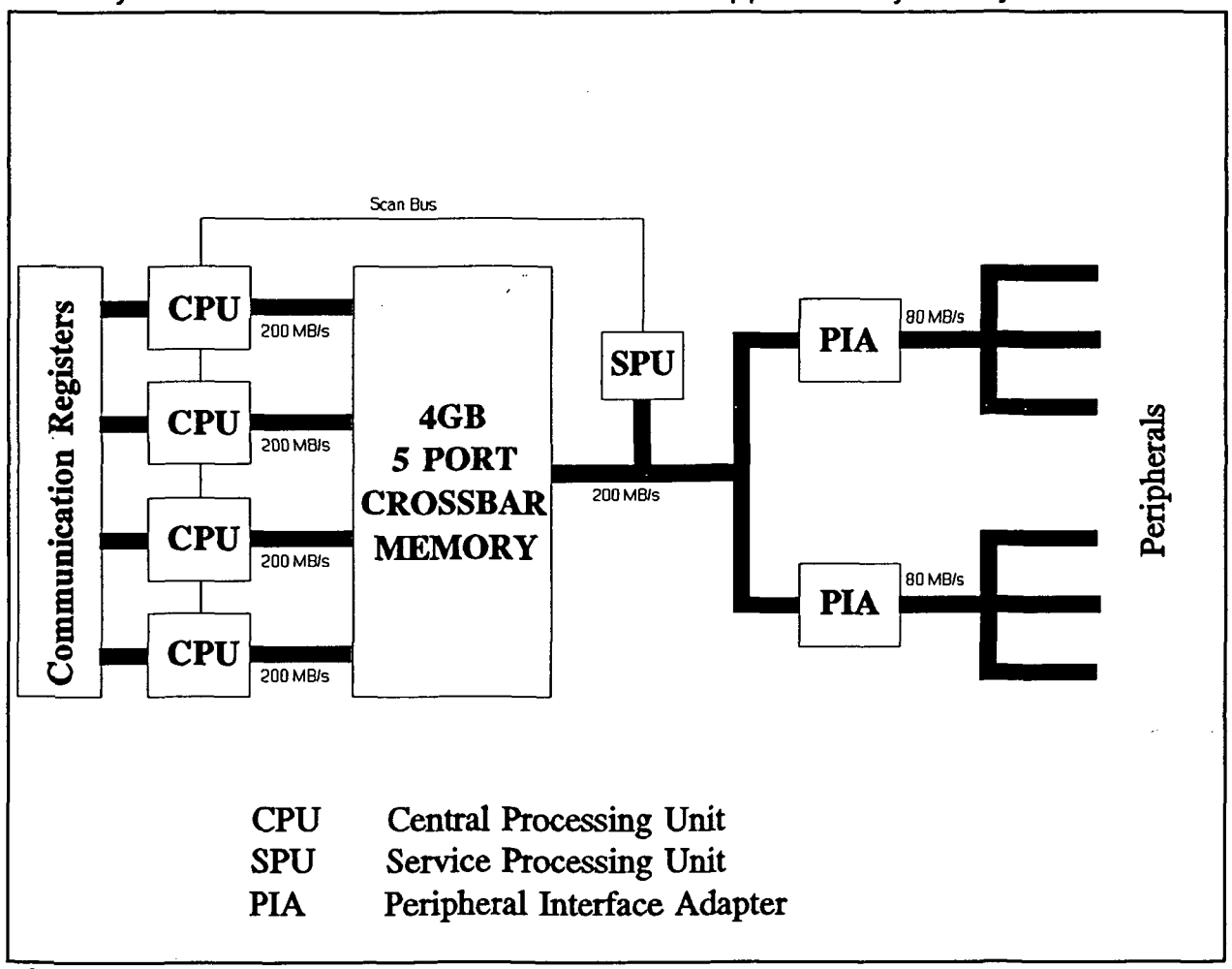


Figure 3 - Convex C-240 Computer System

The C-230i is a specialized Convex system. There are three processors and two I/O modules. The C-2 line can have one to four processing elements and one or two I/O

modules. The total number of major components can not exceed five due to the 5 X 8 crossbar switch. Thus, the 230i system is filled to capacity. This configuration was implemented for a customer that had high I/O bandwidth requirements by Convex's Special Systems Group.

The three processors are capable of 55 MIPS each. They are also capable of vector floating point operations (50 MFLOPS each). A data cache is associated with each processor. It is a write through cache with hardware invalidation between processors for coherency. The cache is bypassed for vector operations.

Parallelism is accomplished in the Convex systems by vectorization of inner loops and parallel execution of outer do-loops along with concurrent execution of independent processes on multiple CPUs. There is a specialized set of communication registers visible to all processing elements where work in progress is posted. If a processor becomes idle, it first checks the communication registers for work already in progress. If there is work in progress that the idle processor can help with, it does so. If no work in progress is posted, the operating system takes over and schedules a new process (ready) to the idle processing element.

The memory system in the Convex computers is arranged in 8 banks. There can be up to 4 Gigabytes of physical memory. However, they are currently shipping systems with only 2 Gigabytes. Memory is connected to I/O and processing elements via a 5 X 8 crossbar or crosspoint switch. The crossbar has 200 Mbytes/sec of bandwidth for each of the 5 access paths.

I/O in the Convex system is memory mapped. The two I/O channels in the C-230i system have 200 Mbytes/sec each. This bandwidth is utilized by two proprietary buses called P-Buses. Each P-Bus is capable of 80 Mbytes/Sec The Special Systems Group of Convex would build any specialized hardware required to interface the 230i to the DIF and to the NASA network.

The Convex computer systems use a flavor of Unix as an operating system. The DHS application software will have to be developed to run on the 230i system. Convex supports the Ada language (validated by Ada Joint Program Office). Convex also supports C and several popular versions of Fortran.

One possible implementation will have a process ready to execute that handles a specific virtual channel. These processes will be multitasked on the three processors and will handle packets as they arrive. Once a packet has been processed, it will be archived and passed along to its end user.

The Convex I/O subsystems are capable or will soon be capable of supporting many standard interfaces. These interfaces include DECnet, TCP/IP, Convex NFS, Network Computing System or NCS, Ultranet, and Hyperchannel. Other interfaces include VMEbus, HiPPI or HSP, IDC, and SNA.

Convex Computer Corporation is closely associated with E-Systems. E-Systems is discussed in another part of this report but the close association between the two companies is beneficial because E-Systems is a likely choice for the database hardware. 1400 Tera Bytes or 1.4 Peta Bytes is approximately the storage required for the DHS and E-Systems has a system on the drawing board that is easily capable of archiving the data.

2.2.2.3.2 Alliant

The Alliant FX/2800 system is a shared memory parallel architecture. The system may contain 28 processors. The processors are based on Intel's i860 RISC microprocessor. This 64 bit processor contains floating point hardware as well as superscalar modes.

The 28 CPU system is capable of 1 GFlops and 1000 VAX MIPS (Dhrystone). This is sufficient processing power. However, the system only has a single I/O module and at least 4 of the processing modules will have to be replaced with I/O modules. A single I/O module can handle 2 channels at 40 MBytes/Sec each. The system can be configured with a variety of modules. Given the appropriate ratio of I/O modules to CPU modules, the FX/2800 system can implement the DHS.

The modules are connected to the main memory via an 8 X 16 crossbar switch capable of 1.28 GBytes/Sec. There are cache memories, connected to memory via a 640 MByte/Sec bus, that can be configured up to 4 MBytes in size. The physical main memory can be 1 GByte in size.

The Alliant line of computers use a UNIX like operating system. Applications for the Alliant can be developed in Fortran, C, or Ada. Also, I/O interfaces such as TCP/IP, DECnet, NFS, NCS, NQS, X-Windows (ver. 11), Ultranet, and VMEbus are supported and available.

2.2.2.3.3 Aptec I/O Bus

The Aptec I/O Bus system is a high performance bus. The system is similar to a bus based, shared memory multiprocessor computer. The difference lies in the fact that none of modules are used for computation. The shared memory of the bus is used for buffering. This memory is very large (1 GByte).

The peak bandwidth of the Aptec bus is 200 Mbytes/Sec. Individual modules can sustain 50 Mbytes/Sec. Thus, a single module can interface to a single DIF 300 Mbps channel. If the utilization of the space link is below 50% the Aptec bus can certainly handle the necessary bandwidth. However, future Aptec products may be capable of peak space link bandwidth.

An Aptec I/O bus requires some processing capabilities to handle the Data Handling Service. One approach is to allow the I/O bus memory to be accessed by a computer system. The data can be manipulated in the shared memory before passing out of the

system. However, this approach involves modifying hardware not intended for this type of application.

2.2.3 Comparisons

The best understood implementation of the Data Handling Service is the Virtual Port Memory Multiprocessor Architecture version. This is due to the extensive simulations. The VPM machine is an excellent choice for functionality. The system is scalable, modular, inexpensive, and flexible. It also degrades gracefully under processor failures. However, this is not a commercially available system.

The Convex system can handle the DHS with some minor hardware interfaces being produced by their Special Applications group. This system is scalable, but its scaling must be done in large steps. Future systems will be able to handle more data than the Space Based Network is capable of producing. The system is modular. The modularity is also coarse. Convex systems are not inexpensive, however, the large software base and customer support is beneficial and non existent with the VPM system. The Convex system is also flexible. The Convex will degrade gracefully but in larger steps.

The other two alternatives presented are less adequate. The Alliant system has more processors than the Convex. However, the bandwidth limitations of the Alliant's I/O system is a fatal flaw for the DHS problem. This may change in the near future. The Aptec I/O bus is incapable of handling the DHS processing requirements and does not appear if it will be able to in the near future.

2.2.4 Issues Requiring Additional Research

Hardware platforms are produced in an engineering fashion. Also, the same platform may be used for any number of applications. However, the software for an application is unique. The DHS is a new application. Thus, the software for the DHS must be developed.

Software development costs are significant for any large system. Anything that can reduce the software development cost is important. A good software development environment is one major asset when producing a software package. The Convex systems have such an environment. The VPM machine does not.

A first generation machine such as the VPM does not have the reliability the older Convex systems do. Also, the VPM machine does not have a well established corporation for production and maintenance. The Convex systems have an extensive customer support staff for consultation on operations.

The enormous bandwidths of the Space Based Network produce a vast quantity of data for archiving. The only system we are aware of that is capable of storing 1400 Tera Bytes of data is produced by E-Systems. The Convex corporation has a close relationship with

E-Systems, and they have worked together in developing the mass storage system. This relationship will ease the interfacing of the DHS system to the mass storage facility.

2.2.5 Recommendations

The Convex system is currently the best choice for a hardware platform for the DHS. The number of processors is increasing and the older system is capable of handling the DHS now. The I/O bandwidth of the Convex system is adequate and increasing.

The software development costs will be enormous and the development environment on the Convex platform will ease this development. The interface between the DHS and storage system is already established. Finally, the large body of software running on the Convex systems is a great asset.

References

1. Computer Sciences Corporation, Space Station Era Processing Function Architecture Matching Report, Computer Sciences Corporation, July 1989.

2. Computer Sciences Corporation, Space Station Era Telemetry Processing Identification Report, Computer Sciences Corporation, March 1989.

3. Polson, John T. "Design of Advanced Space Data Handling Service Using Virtual Port Memory Multiprocessor Architecture" Ms. thesis, New Mexico State University, 1990.

4. Polson, John T., "A Parallel Computer Approach for Processing Space Station Telemetry Packets," Proceedings International Telemetering Conference 1990, Las Vegas, Nevada.

5. Johnson, Eric, "The Virtual Port Memory Multiprocessor Architecture," Ph.D. dissertation, New Mexico State University, 1987.

6. Johnson, Eric, "ARGOS - A Research GMMP Operating System: Overview and Interfaces," New Mexico State University, Aug. 1989.

7. CACI, NETWORK II.5 User's Manual, CACI, 1988.

8. Cheung, Shun et al., Introduction to Simulation Using Network II.5, CACI, 1988.

9. Lazowska, E.D. et al., Quantitative System Performance Computer System Analysis Using Queueing Network Models. New Jersey: Prentice-Hall, Inc., 1984.

10. Horan, S., Pfeiffer, J., and Taylor, J., SEMI-ANNUAL REPORT COOPERATIVE RESEARCH AGREEMENT TELEMETRY DOWNLINK INTERFACES AND LEVEL-ZERO PROCESSING, Period Covered: 1 October 1989 - 30 April 1990, Electrical and Computer Engineering Department, New Mexico State University.

11. Horan, S., Pfeiffer, J., and Taylor, J., SEMI-ANNUAL REPORT COOPERATIVE RESEARCH AGREEMENT TELEMETRY DOWNLINK INTERFACES AND LEVEL-ZERO PROCESSING, Period Covered: 1 May 1990 - 30 September 1990, Electrical and Computer Engineering Department, New Mexico State University.

12. Horan, S., Pfeiffer, J., and Taylor, J., SEMI-ANNUAL REPORT COOPERATIVE RESEARCH AGREEMENT TELEMETRY DOWNLINK INTERFACES AND LEVEL-ZERO PROCESSING, Period Covered: 1 October 1990 - 30 April 1991, Electrical and Computer Engineering Department, New Mexico State University.

2.3 NETWORK II.5 DATA INTERFACE FACILITY (DIF) MODEL (A. K. Petersen)

Construction of a Network II.5 simulation model of the Data Interface Facility (DIF) was accomplished. The model is logically partitioned into three segments: creation, reception, and transmission.

The creation segment simulates the TDRSS by creating data packets. The data is composed of two types: high and low rate traffic. Low rate packets represent normal traffic, and constitute a steady stream of information. High rate packets (often referred to as burst rate packets) arrive much less frequently, but require almost 100% of the space channel for several seconds. The current burst rate is represented as a 3 second burst, every 5 minutes, at 85% capacity. Taken together, these two classes of traffic utilize approximately 10% of the available down-link bandwidth. An average of 1% of the arriving packets are modeled as being garbled.

Approximately every 90 minutes, the communications channel is interrupted as the Space Station passes out of contact with the TDRSS. During this Zone Of Exclusion (ZOE) the spacecraft must queue it's Earth bound traffic. This loss of signal lasts from 0 to 14 minutes, with a mean at 7. Once the signal is reacquired, the queued data (and real time data) are dumped to Earth at the capacity of the space channel. This priority playback traffic is the driving force behind the amount of buffering needed in the Data Interface Facility.

The reception segment of the model simulated the front end of the DIF. At this point, data packets are read from the space channel and queued in the buffer.

The third segment of the DIF model is the out link transmission to the DHC. This part of the model was separated from the other functions because the exact nature of the data link between the DIF and the DHC has not been decided. In the best case scenario, the DHC is located physically near the DIF, and a fiber optic land line will connect them. This communication medium can keep up with the space channel data rate and, therefor, eliminates the need for a buffer.

However, the possibility exists that the DIF and the DHC will be separated by thousands of miles. A communications satellite is the most likely link between the two. Use of an existing DOMSAT has been purposed. Since the bandwidth of this satellite is only 50 Mhz, a large buffer will be needed to average and match data rates from the 300 MHz TDRSS link.

During the modeling of the reception segment of the DIF simulation, one problem has been found: Network II.5. Originally the model required 52 hours to simulate one hour of real time. Much of this time was spent simulating operations which Network II.5 either did not support directly, or worse, supported incorrectly (particularly semaphores). Two upgrades to Network II.5 have become available since this task was undertaken. The latest version appears to have solve many of these problems, allowing the simulation to be re-written in a more straight forward fashion. As a result, the simulated-to-real time

ratio has fallen to 6:1. Six to one is still not very good if several hours of time are to be simulated, as is the case to fully analyze the effects of several ZOE's.

A second major problem is the Sun 4/280 has a mean-time-between-failure (MTBF) rate of about 6 days. As a result, only half the simulation runs ever complete. The amount of usable data obtained is, thus far, meager. The few results currently available are found under "RESULTS", below. Special complaints should be made to CACI concerning the limited number of digits available for results in the printouts. This model overflows the field after only 45 minutes of simulated time.

With respect to the DIF simulation effort, effective simulation requires either using a faster (which probably means lower level) simulator, like SimScript, or access to a faster computer (like a Cray).

The complete Network II.5 model has been included in Section 2.3.3.

2.3.1 GPSS/H DIF Model

In an attempt to overcome Network II.5 problems (outlined above), several other simulators were evaluated. For a variety of reasons, Wolverine Software's GPSS/H was chosen. A model of the DIF has been written (see Section 2.3.4) to parallel the Network II.5 version.

GPSS/H is currently running on a SUN 3 with a simulation-to-computer time ratio of approximately 70:1. Although this does not seem to be much better than Network II.5, two factors must be considered when trying to compare run times. Network II.5 reports connect time, while GPSS/H reports CPU time. Thus Network II.5's simulation times are influenced by the load on the machine, GPSS/H's figures are not. Secondly, GPSS/H is running on a much less powerful platform. We have access to GPSS/H on an IBM ES-9000, which should be remarkably faster.

GPSS/H is a transaction based simulator, as opposed to Network II.5's event driven simulation. This difference results in a radical change in the appearance of the model when translating into GPSS/H. Additionally, GPSS/H handles it's random number generators in a manner which makes use of the normal distribution impractical for time values. Thus, normal distributions in Network II.5 have been changed to triangular in GPSS/H. These two differences between simulators produces slight , but noticeable, differences in the results. Therefore, care should be exercised when comparing the two sets of results.

2.3.2 Results

A steady state analysis of the DIF reveals the DOMSAT link will be approximately 62% utilized at the beginning of the Space Station Freedom project. This assumes only 10% of the available TDRSS bandwidth will be used. There is not much room to expand. This

bottleneck would, of course, be eliminated if the DIF and DHC where within fiber-optic-link distance of each other.

More importantly, though, is the delay experienced by the data packets during high rate traffic periods. This is at it's worst immediately following a ZOE. All the priority playback data, as well as the normal data will be sent to earth at 300 Mb/s, thus overwhelming the DOMSAT's 50 Mb/s capacity. The amount of memory required to buffer this data, and the delay involved are two questions which these models can answer.

The results available to date indicate 16,000 packets (16 MB) will need to be buffered. An arbitrary packet should not expect to have to wait more than 250 microseconds in the buffer. Note this delay value does not take into account retransmission requests from the DHC as the result of DOMSAT corruption.

2.3.2 Recommendations

The Network II.5 model should be abandoned, unless a super computer can be found. Even then, the Network II.5 model should be used only as a sanity check on a more efficient simulator. For the moment, GPSS/H seems to be satisfactory. The model needs to be run long enough to simulate 1-2 days of real time. A way needs to be found to pipe the DOMSAT traffic to a model of the DHC as it's input.

2.3.3 Network II.5 Data Interface Facility Model Script

The Network II.5 model is commented as much as Network allows, but additional notes may be helpful. A Network II.5 model consists of a collection of entities. There is no significance to the order or placement of the entity definitions in the script, as the behavior of each is completely defined by it's definition. This feature makes a Network II.5 script difficult to understand and visualize.

Conceptually, a flow chart would be the most logical way to describe a Network II.5 model, but the complexity of describing all the possible mutual dependencies make the flow chart as hard to read as the original script.

The entities are divided into classes as follows:

Global Variables: Global variables act as compiler directives, allowing batch mode operation. They are fairly self-explanatory.

6 Statistical Distributions: Each statistical distribution function (SDF) defines the type of distribution, the bounds, and a random number stream.

LOW RATE ITERATION How often a low rate burst occurs. Low rate bursts simulate normal, background, traffic. Examples include space station status, health and welfare, voice, etc. The mean value was chosen to obtain the desired TDRSS utilization of 10%.

LOW RATE DURATION How many packets (1024 bytes each) are in each low rate burst.

HIGH RATE ITERATION How often a high rate burst occurs. High rate bursts simulate the transfer of large amounts of data. Examples include instrument data and video.

HIGH RATE DURATION How many packets (1024 bytes each) are in each high rate burst.

ZOE ITERATION How often a Zone of Exclusion is encountered. The ZOE results from the space station being out of contact with a TDRSS.

ZOE DURATION How long each Zone of exclusion lasts, measured in TDRSS packet frames.

4 Processing Elements: Each processing element (PE) has an individual clock speed and instruction set.

SPACE STATION This PE generates the low and high rate data packets. The clock speed is arbitrary but was chosen to make specification of the data packet iteration periods convenient. Each data packets is represented by a count of the semaphore INSTRUMENT DATA.

TDRSS This PE represents the downlink from the space station to the DIF. The clock speed is equal to the time needed to transmit one data packet (1024 bytes + 32 sync bits).

DIF This PE represents the data interface facility. The packets are checked for errors here. Bad packets are discarded and a retransmission is requested. As with the Space Station, the clock speed was chosen for convenience. Received packets are represented as counts of the semaphore RAW DATA. Checked (good) packets are represented by counts of the semaphore PROCESSED DATA.

DOMSAT This PE represents the uplink from the DIF to the DHC. The clock speed is equal to the time needed to transmit one data packet (1024 bytes + 32 sync bits). Transmitted data packets are represented by counts of the semaphore USER DATA.

10 Software Modules:

Each software module represents a program or task which a processor must execute. Modules may be started by a wide variety of combinations of circumstances. Once started, a module directs the PE to execute a series of instructions. Upon completion, a module may call (chain to) another module.

INITIALIZE SPACE STATION Executes only once, at the beginning of the simulation on PE SPACE STATION. Resets the INSTRUMENT DATA semaphore.

CREATE LOW RATE VCDU'S Executes every LOW RATE ITERATION microseconds, on PE SPACE STATION. Increments the INSTRUMENT DATA semaphore by LOW RATE DURATION.

CREATE HIGH RATE VCDU'S Executes every HIGH RATE ITERATION microseconds, on PE SPACE STATION. Increments the INSTRUMENT DATA semaphore by HIGH RATE DURATION.

INITIALIZE TDRSS Executes only once, at the beginning of the simulation, on PE TDRSS. Resets the RAW DATA semaphore.

TRANSMIT ONE VCDU VIA TRDSS Executes anytime the INSTRUMENT DATA semaphore has a count greater than 0, on PE TDRSS. Decrements the INSTRUMENT DATA semaphore, waits for one TDRSS packet time, then increments the RAW DATA semaphore. Chains to itself to be ready for the next packet.

ZONE OF EXCLUSION Executes every ZOE ITERATION microseconds, on PE TDRSS. The high priority of this module allows it to take control of PE TDRSS for ZOE DURATION packet frames. While this module is in control, module TRANSMIT ONE VCDU VIA TDRSS (see above) cannot execute, simulating a break in the downlink.

INITIALIZE DIF Executes only once, at the beginning of the simulation, on PE DIF. Resets the PROCESSED DATA semaphore.

SANITY CHECK Executes anytime the RAW DATA semaphore has a count greater than 0, on PE DIF. Decrements the RAW DATA semaphore, waits long enough to check the data packet for errors. 99.9% of the time, this module increments the PROCESSED DATA semaphore to simulate an error free (good) packet. 0.01% of the time this module increments the INSTRUMENT DATA semaphore to simulate a request for retransmission of a bad data packet. Chains to itself to be ready for the next packet.

INITIALIZE DOMSAT Executes only once, at the beginning of the simulation, on PE DOMSAT. Resets the USER DATA semaphore.

TRANSMIT ONE VCDU VIA DOMSAT Executes anytime the PROCESSED DATA semaphore has a count greater than 0, on PE TDRSS. Decrements the PROCESSED DATA semaphore, waits for one DOMSAT packet time, then increments the USER DATA semaphore. Chains to itself to be ready for the next packet.

1 Instruction Mix: Each instruction mix (IM) is translated into one of several instructions, based on assigned probabilities.

HANDLE ONE RAW VCDU This mix determines whether a data packet received at the DIF is good or bad.

1 Macro Instruction:

Each macro instruction (MI) expands into a number of instructions, allowing instructions which are commonly used together to be collected and referred to by a single name.

GENERATE ONE PACKET Used to create the individual packets of a low or high rate burst. Waits some time for the packet to be generated, then increments the INSTRUMENT DATA semaphore.

4 Semaphores:

Each semaphore is a counter which can never be decremented below zero. The value of the semaphore can be changed in zero time. Semaphores may be incremented, decremented, or set to an arbitrary value.

INSTRUMENT DATA Represents the data packets, onboard the Space Station, waiting to be sent to Earth. Reset (set to zero) by INITIALIZE SPACE STATION. Incremented by CREATE LOW RATE VCDU'S, CREATE HIGH RATE VCDU'S, and SANITY CHECK (bad packets). Decremented by TRANSMIT ONE VCDU VIA TDRSS.

RAW DATA Represents the data packets, received by the DIF, waiting to be error checked. Reset (set to zero) by INITIALIZE TDRSS. Incremented by TRANSMIT ONE VCDU VIA TDRSS. Decremented by SANITY CHECK (good packets).

PROCESSED DATA Represents the data packets, in the DIF, waiting to be sent to the DHC. Reset (set to zero) by INITIALIZE DIF. Incremented by SANITY CHECK (good packets). Decremented by TRANSMIT ONE VCDU VIA DOMSAT.

USER DATA Represents the data packets, received by the DHC. Reset (set to zero) by INITIALIZE DOMSAT. Incremented by TRANSMIT ONE VCDU VIA DOMSAT.

Beginning on the next page is the actual Network II.5 script. Comments are introduced by an asterisk (*) in column 1.

```
* SSIS DIF V.10
*
*    NETWORK II.5 SIMULATION SCRIPT
*
*    SPACE STATION INFORMATION SYSTEM (SSIS)
*    DATA INTERFACE FACILITY (DIF)
*
*    A.K. PETERSEN, NEW MEXICO STATE UNIVERSITY
*
*
***** NETIN RELEASE 6.03  THIS FILE SAVED 06/13/1991 15:13:17

***** GLOBAL VARIABLES
GLOBAL FLAGS =
  ANTITHETIC VARIATE = NO
  RANDOMIZER =  2
  MINIMIZE RANDOM SEED ARRAY = NO
  NETIN TIME UNITS = SECONDS
  ITERATE BY PRIORITY = YES
  CLOCK = YES
  BATCH = YES
  INPUT LISTING = YES
  DEFAULT LISTING = NO
  LENGTH = 7200.0 SECONDS
  RUNTIME WARNINGS = TERMINAL
  PERIODIC REPORTS =  24
    SEMAPHORE
    SNAPSHOT
    INSTRUCTION
  PLOT DATA FILE = NO
  WIDE REPORTS = NO
  TRACE = NO

***** STATISTICAL DISTRIBUTION FUNCTIONS
STATISTICAL DISTRIBUTIONS =
*
* HOW OFTEN A LOW RATE BURST OCCURS
* UNITS ARE MICROSECONDS
* CURRENT MEAN: 2666.66667 PACKETS/SECOND
*
  NAME = LOW RATE ITERATION
    TYPE = NORMAL
    MEAN =      375.000000
    STANDARD.DEVIATION =       25.000000
    LOWER.BOUND =       0.
    UPPER.BOUND =     1000.000000
```

```
   STREAM =    1
*
* HOW MANY VCDU'S ARE SENT IN LOW RATE BURST
* UNITS ARE PACKETS PER BURST
* CURRENT MEAN: 1 PACKET
*
  NAME = LOW RATE DURATION
    TYPE = EXPONENTIAL
    MEAN =        1.000000
    LOWER.BOUND =        1.000000
    UPPER.BOUND =        8.000000
    STREAM =    2
*
* HOW OFTEN HIGH RATE BURSTS OCCUR
* UNITS ARE MICROSECONDS
* CURRENT MEAN: 5 MINUTES
*
  NAME = HIGH RATE ITERATION
    TYPE = NORMAL
    MEAN =  +3.000000E+008
    STANDARD.DEVIATION = 15000000.000000
    LOWER.BOUND =        0.
    UPPER.BOUND =  +5.000000E+008
    STREAM =    3
*
* HOW LONG A HIGH RATE BURST LASTS
* UNITS ARE CPU CYCLES
* CURRENT MEAN: 2.5 SECONDS
*
  NAME = HIGH RATE DURATION
    TYPE = NORMAL
    MEAN =    15200.000000
    STANDARD.DEVIATION =     1000.000000
    LOWER.BOUND =        0.
    UPPER.BOUND =    20000.000000
    STREAM =    4
*
* HOW OFTEN THE ZONE OF EXCLUSION OCCURS
* UNITS ARE MICROSECONDS
* CURRENT MEAN: 90 MINUTES
*
  NAME = ZOE ITERATION
    TYPE = NORMAL
    MEAN =  +5.400000E+009
    STANDARD.DEVIATION = 10000000.000000
    LOWER.BOUND =  +4.860000E+009
```

```
   UPPER.BOUND =  +5.940000E+009
   STREAM =    5
*
* HOW LONG THE ZONE OF EXCLUSION LASTS
* UNITS ARE SECONDS
* CURRENT MEAN: 7 MINUTES
*
  NAME = ZOE DURATION
    TYPE = NORMAL
    MEAN =       420.000000
    STANDARD.DEVIATION =       42.000000
    LOWER.BOUND =        0.
    UPPER.BOUND =      840.000000
    STREAM =    6

***** PROCESSING ELEMENTS
HARDWARE TYPE = PROCESSING
*
* ORBITING SPACE STATION
* ONE CPU CYCLE = ONE MICROSECOND
*
  NAME = SPACE STATION
    BASIC CYCLE TIME =      1.000000 MICROSEC
    INPUT CONTROLLER = NO
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = PROCESSING
        NAME ; DATA GENERATION
          TIME ; 1 CYCLES
      INSTRUCTION TYPE = SEMAPHORE
        NAME ; FLUSH INSTRUMENT DATA
          SEMAPHORE ; INSTRUMENT DATA
          SET/RESET FLAG ; RESET
          EQUAL TO ;  0
        NAME ; INCREMENT INSTRUMENT DATA
          SEMAPHORE ; INSTRUMENT DATA
          SET/RESET FLAG ; SET
*
* TDRSS SATELLITE DOWN-LINK
* ONE CPU CYCLE = ONE TDRSS FRAME TIME
*
  NAME = TDRSS
    BASIC CYCLE TIME =     27.413330 MICROSEC
    INPUT CONTROLLER = NO
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = PROCESSING
        NAME ; TDRSS ACCESS
```

```
        TIME ; 1 CYCLES
      NAME ; ZOE
        TIME ; ZOE DURATION
    INSTRUCTION TYPE = SEMAPHORE
      NAME ; FLUSH RAW DATA
        SEMAPHORE ; RAW DATA
        SET/RESET FLAG ; RESET
        EQUAL TO ;  0
      NAME ; INCREMENT RAW DATA
        SEMAPHORE ; RAW DATA
        SET/RESET FLAG ; SET
      NAME ; DECREMENT INSTRUMENT DATA
        SEMAPHORE ; INSTRUMENT DATA
        SET/RESET FLAG ; RESET
*
* DATA INTERFACE FACILITY
* ONE CPU CYCLE = ONE MICROSECOND
*
  NAME = DIF
    BASIC CYCLE TIME =      1.000000 MICROSEC
    INPUT CONTROLLER = NO
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = PROCESSING
        NAME ; ERROR CHECK
          TIME ; 1 CYCLES
      INSTRUCTION TYPE = SEMAPHORE
        NAME ; FLUSH PROCESSED DATA
          SEMAPHORE ; PROCESSED DATA
          SET/RESET FLAG ; RESET
          EQUAL TO ;  0
        NAME ; INCREMENT PROCESSED DATA
          SEMAPHORE ; PROCESSED DATA
          SET/RESET FLAG ; SET
        NAME ; INCREMENT INSTRUMENT DATA
          SEMAPHORE ; INSTRUMENT DATA
          SET/RESET FLAG ; SET
        NAME ; DECREMENT RAW DATA
          SEMAPHORE ; RAW DATA
          SET/RESET FLAG ; RESET
*
* TRANSMIT DATA PACKETS OVER THE DOMSAT LINK
* ONE CPU CYCLE = ONE DOMSAT FRAME TIME
*
  NAME = DOMSAT
    BASIC CYCLE TIME =    164.480000 MICROSEC
    INPUT CONTROLLER = YES
```

```
    INSTRUCTION REPERTOIRE =
      INSTRUCTION TYPE = PROCESSING
        NAME ; DOMSAT ACCESS
          TIME ; 1 CYCLES
      INSTRUCTION TYPE = SEMAPHORE
        NAME ; FLUSH USER DATA
          SEMAPHORE ; USER DATA
          SET/RESET FLAG ; RESET
          EQUAL TO ;  0
        NAME ; INCREMENT USER DATA
          SEMAPHORE ; USER DATA
          SET/RESET FLAG ; SET
        NAME ; DECREMENT USER DATA
          SEMAPHORE ; USER DATA
          SET/RESET FLAG ; RESET
        NAME ; DECREMENT PROCESSED DATA
          SEMAPHORE ; PROCESSED DATA
          SET/RESET FLAG ; RESET

***** MODULES
SOFTWARE TYPE = MODULE
*
*
* INITIALIZE SPACE STATION
* EXECUTES ONLY ONCE, AT THE START OF A SIMULATION
*
  NAME = INITIALIZE SPACE STATION
    PRIORITY =        100
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    START TIME = 0.0
    ALLOWED PROCESSORS =
      SPACE STATION
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 FLUSH INSTRUMENT DATA
*
* CREATE LOW RATE VCDU'S TO SIMULATE NORMAL TRAFFIC
*
  NAME = CREATE LOW RATE VCDU'S
    PRIORITY =         0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = YES
    ITERATION PERIOD = LOW RATE ITERATION
    ALLOWED PROCESSORS =
      SPACE STATION
    INSTRUCTION LIST =
```

```
    NUMBER OF INSTRUCTIONS ; LOW RATE DURATION
    INSTRUCTION NAME ; GENERATE ONE PACKET
*
* CREATE HIGH RATE VCDU'S TO SIMULATE BURST TRAFFIC
*
 NAME = CREATE HIGH RATE VCDU'S
  PRIORITY =          1
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = YES
  ITERATION PERIOD = HIGH RATE ITERATION
  ALLOWED PROCESSORS =
   SPACE STATION
  INSTRUCTION LIST =
    NUMBER OF INSTRUCTIONS ; HIGH RATE DURATION
    INSTRUCTION NAME ; GENERATE ONE PACKET
*
* INITIALIZE TDRSS LINK
* EXECUTES ONLY ONCE, AT THE START OF A SIMULATION
*
 NAME = INITIALIZE TDRSS
  PRIORITY =        100
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  START TIME = 0.0
  ALLOWED PROCESSORS =
   TDRSS
  INSTRUCTION LIST =
   EXECUTE A TOTAL OF ; 1 FLUSH RAW DATA
*
* TRANSMIT VCDU'S TO DIF
*
 NAME = TRANSMIT ONE VCDU VIA TDRSS
  PRIORITY =          0
  INTERRUPTABILITY FLAG = YES
  CONCURRENT EXECUTION = YES
  START TIME = 0.0
  ALLOWED PROCESSORS =
   TDRSS
  REQUIRED SEMAPHORE STATUS =
   RUN WHEN ; INSTRUMENT DATA
   IS ; > 0
  INSTRUCTION LIST =
   EXECUTE A TOTAL OF ; 1 TDRSS ACCESS
   EXECUTE A TOTAL OF ; 1 DECREMENT INSTRUMENT DATA
   EXECUTE A TOTAL OF ; 1 INCREMENT RAW DATA
  ANDED SUCCESSORS =
```

```
    CHAIN TO ; TRANSMIT ONE VCDU VIA TDRSS
    WITH ITERATIONS THEN CHAIN COUNT OF ;      0
*
* BLOCK LINK TO EARTH DURING ZONE OF EXCLUSION
*
 NAME = ZONE OF EXCLUSION
  PRIORITY =          10
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  ITERATION PERIOD = ZOE ITERATION
  ALLOWED PROCESSORS =
   TDRSS
  INSTRUCTION LIST =
   EXECUTE A TOTAL OF ; 1 ZOE
*
* INITIALIZE DATA INTERFACE FACILITY
* EXECUTES ONLY ONCE, AT THE START OF A SIMULATION
*
 NAME = INITIALIZE DIF
  PRIORITY =         100
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = NO
  START TIME = 0.0
  ALLOWED PROCESSORS =
   DIF
  INSTRUCTION LIST =
   EXECUTE A TOTAL OF ; 1 FLUSH PROCESSED DATA
*
* CHECK DATA PACKET FOR ERRORS
*
 NAME = SANITY CHECK
  PRIORITY =          0
  INTERRUPTABILITY FLAG = NO
  CONCURRENT EXECUTION = YES
  START TIME = 0.0
  ALLOWED PROCESSORS =
   DIF
  REQUIRED SEMAPHORE STATUS =
   RUN WHEN ; RAW DATA
   IS ; > 0
  INSTRUCTION LIST =
   EXECUTE A TOTAL OF ; 1 ERROR CHECK
   EXECUTE A TOTAL OF ; 1 DECREMENT RAW DATA
   EXECUTE A TOTAL OF ; 1 HANDLE ONE RAW VCDU
  ANDED SUCCESSORS =
   CHAIN TO ; SANITY CHECK
```

```
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0
*
* INITIALIZE DOMSAT
* EXECUTES ONLY ONCE, AT THE START OF A SIMULATION
*
  NAME = INITIALIZE DOMSAT
    PRIORITY =        100
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = NO
    START TIME = 0.0
    ALLOWED PROCESSORS =
      DOMSAT
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 FLUSH USER DATA
*
* TRANSMIT VCDU'S TO USERS
*
  NAME = TRANSMIT ONE VCDU VIA DOMSAT
    PRIORITY =         0
    INTERRUPTABILITY FLAG = NO
    CONCURRENT EXECUTION = YES
    START TIME = 0.0
    ALLOWED PROCESSORS =
      DOMSAT
    REQUIRED SEMAPHORE STATUS =
      RUN WHEN ; PROCESSED DATA
      IS ; > 0
    INSTRUCTION LIST =
      EXECUTE A TOTAL OF ; 1 DECREMENT PROCESSED DATA
      EXECUTE A TOTAL OF ; 1 INCREMENT USER DATA
      EXECUTE A TOTAL OF ; 1 DOMSAT ACCESS
      EXECUTE A TOTAL OF ; 1 DECREMENT USER DATA
    ANDED SUCCESSORS =
      CHAIN TO ; TRANSMIT ONE VCDU VIA DOMSAT
      WITH ITERATIONS THEN CHAIN COUNT OF ;      0

***** INSTRUCTION MIXES
SOFTWARE TYPE = INSTRUCTION MIX
*CREATE A MIXTURE OF GOOD AND BAD VCDU'S
  NAME = HANDLE ONE RAW VCDU
    INSTRUCTIONS ARE ;  99.9000 % INCREMENT PROCESSED DATA
    INSTRUCTIONS ARE ;    .1000 % INCREMENT INSTRUMENT DATA
    STREAM = 101

***** MACRO INSTRUCTIONS
SOFTWARE TYPE = MACRO INSTRUCTION
```

```
*
*
* CREATE A DATA PACKET
*
  NAME = GENERATE ONE PACKET
    NUMBER OF INSTRUCTIONS ; 1
    INSTRUCTION NAME ; DATA GENERATION
    NUMBER OF INSTRUCTIONS ; 1
    INSTRUCTION NAME ; INCREMENT INSTRUMENT DATA

***** SEMAPHORES
SOFTWARE TYPE = SEMAPHORE
  NAME = INSTRUMENT DATA
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES = 2147283647
  NAME = RAW DATA
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES = 2147283647
  NAME = PROCESSED DATA
    INHIBIT RESPONSE = YES
    MAXIMUM PENDING RESPONSES = 2147283647
  NAME = USER DATA
    INHIBIT RESPONSE = NO
    MAXIMUM PENDING RESPONSES = 2147283647
```

2.3.4 GPSS/H Data Interface Facility Model Script

The GPSS/H model is commented much more extensively than the Network II.5 version. In addition, GPSS/H comes closer to a conventional programming language, and is therefor easier to read, understand and interpret.

GPSS/H is a transaction based simulator. Data structures called transactions are created by GENERATE statements (called BLOCKS in the GPSS/H scheme of things). Transactions are then acted upon by a succession of blocks (statements) in the order listed by the script (program). The transactions can be though of as 'traveling' through the model.

The simulation script can be logically divided into sections (analogous to routines in a conventional programming language). These sections are outlined below:

Symbol Declarations: The character or characters with which a symbol name begins define the type of symbol:

&	Global variable. Declared to either Integer, Real, Character, or External.
PB$	Transaction byte parameter.
PH$	Transaction half-word parameter.
PF$	Transaction full-word parameter.
PL$	Transaction floating-point parameter.
RV	Built-in random variate generator. RVTRI has a triangular distribution, RVEXPO an exponential, RVNORM a normal, etc.
FN$	User defined functions.

All other symbolic names are constants used by the compiler for value substitution at compile time.

Note that members of each class of object (queues, storages, logic switches, etc) are referred to by number, QUEUE 1, STORAGE 5, etc. Objects referenced by symbolic name are automatically assigned numbers in the order the compiler discovers them. The EQU directive allows the programmer to explicitly define the name-to-number relationship in advance.

Simulation Control: The SIMULATE statement informs the compiler this script is to be compiled and then run.

Termination Transaction: A single transaction is generated at time &LENGTH which is immediately terminated with a count of one (1) to end the simulation.

Low Rate Data Packets: A single transaction is generated every LRITAVG microseconds (on average) which is marked as being low rate, then split into FN$LRDUR identical transactions to represent a low rate burst. Each of these then enter the low rate queue and, if the model is currently in a zone of exclusion, the transactions are marked as such and enter the ZOEWAIT queue. Finally, the low rate packets are transferred to the TDRSS downlink.

High Rate Data Packets: A single transaction is generated every HRITAVG microseconds (on average) which is marked as being high rate, then split into HRDUR identical transactions to represent a high rate burst. Each of these then enter the high rate queue and, if the model is currently in a zone of exclusion, the transactions are marked as such and enter the ZOEWAIT queue. Finally, the high rate packets are transferred to the TDRSS downlink.

High rate packets have a higher priority than low rate packets, so they proceed through the model more quickly.

Zone Of Exclusion: A single transaction is generated every ZEITAVG microseconds (on average) which enter the ZOEQUE queue, declares a zone of exclusion, and capture the TRDSS downlink for ZEDURAVG microseconds. The zone of exclusion transactions have a higher priority than either high or low rate data packets, so the TDRSS downlink is blocked immediately. After the prescribed amount of time, the transaction releases the TDRSS downlink, leaves the queue, and terminates with no count (so the simulation can continue).

TDRSS Downlink: Low and high rate data packets enter the TDRSQUE queue and proceed to capture the TDRSS downlink. The transaction departs either LRQUE or HRQUE (depending on how the transaction was flagged at birth), then holds the TDRSS downlink for the proper frame time. A logic switch is used to insure no two packets are using the downlink at the same time. Once the packet has been 'transmitted', it releases the TDRSS downlink, departs the ZOEQUE queue, and moves on to the Data Interface Facility.

Data Interface Facility: Data packets arrive at the DIF, enter the DIFQUE queue, and wait a period of time to simulate error checking. The packets then leave the DIFQUE queue and either move on to the DOMSAT uplink (good packets) or to Request Retransmission (bad packets). Bad packets occur every RETRYAVG packets (on average).

Request Retransmission: Bad data packets are placed back in LRQUE or HRQUE (depending on how they were flagged at birth), and 'returned' to the TDRSS downlink.

DOMSAT Uplink: All good packets are laced in the DOMQUE queue and pass sequentially through the DOMSAT logic gate as described for the TDRSS downlink. Once through the DOMSAT uplink, transactions depart the DOMQUE queue and terminate with no count (so the simulation can continue).

Control Statements: Control statements are not part of the model, per se, but do effect the way in which it runs. Sufficient memory is set aside for COMMON storage (transactions, queues, etc), logic gates are initialized, and the simulation started. The START statement defines how many transaction termination counts are required to end the simulation. Since only the termination transaction has a non-zero count (low rate, high rate, and zone of exclusion transactions terminate with no counts) only one count is specified in the START statement.

The major objects used in the simulation are described below:

Queues:

LRQUE Collects statistics on how long a low rate data packet must wait in the space station before being sent to the DIF (including ZOE delays).

HRQUE Collects statistics on how long a high rate data packet must wait in the space station before being sent to the DIF (including ZOE delays).

ZOEQUE Collects statistics on how long the zones of exclusion last.

ZOEWAIT Collects statistics on how long packets (both low and high rate) must wait in the space station as the result of a zone of exclusion.

TDRSQUE Collects statistics on how long packets (both low and high rate) must wait in the space station before being sent to the DIF (including ZOE delays).

DIFQUE Collects statistics on how long packets (both low and high rate) take to move through the DIF.

DOMQUE Collects statistics on how long packets (both low and high rate) must wait in the DIF before gaining access to the DOMSAT uplink.

Transaction Parameters:

BIRTHQUE Holds the number associated with the queue to which a transaction is assigned at birth. For low rate transactions BIRTHQUE=LRQUE, for high rate transactions BIRTHQUE=HRQUE.

LONGWAIT Holds a flag indicating whether or not this transaction had to wait for a zone of exclusion.

0	No wait was encountered.
1	A wait was encountered.

Logic Switches:

TDRSS Used to control access to the TDRSS downlink.

RESET	The downlink is available for use.
SET	The downlink is currently in use.

DOMSAT Used to control access to the DOMSAT uplink.

RESET	The uplink is available for use.
SET	The uplink is currently in use.

ZONE Used to flag zones of exclusion.

RESET	The space station is current in contact with the DIF, via TRDSS.
SET	The space station is in a zone of exclusion.

Beginning on the next page is the actual GPSS/H script. Comments are introduced by an asterisk (*) in column 1, or text occurring after any arguments.

```
*************************************************************************
*                                                                       *
*          SSIS DIF V.2                                                 *
*                                                                       *
*          GPSS/H SIMULATION SCRIPT                                     *
*                                                                       *
*          SPACE STATION INFORMATION SYSTEM (SSIS)                      *
*          DATA INTERFACE FACILITY (DIF)                                *
*                                                                       *
*          A. K. PETERSEN, NEW MEXICO STATE UNIVERSITY                  *
*          JULY, 1991                                                   *
*                                                                       *
*************************************************************************
*
***** FUNCTION DECLARATIONS
*
LRDUR     FUNCTION  RN(100),D8                NUMBER OF PACKETS PER
.632,0/.865,1/.950,2/.982,3/.993,4/.997,5/.999,6/1.00,8 LOW RATE BURST

*
***** ENTITY EQUATES
*
*                                             QUEUES:
LRQUE     EQU       1,Q                             LOW RATE DATA
HRQUE     EQU       2,Q                             HIGH RATE DATA
ZOEQUE    EQU       3,Q                             ZONE OF EXCLUSION
ZOEWAIT   EQU       4,Q                             WAIT FOR ZOE
TDRSQUE   EQU       5,Q                             INSTRUMENT DATA
DIFQUE    EQU       6,Q                             PROCESSED DATA
DOMQUE    EQU       7,Q                             USER DATA

*                                             TRANSACTION PARAMETERS:
BIRTHQUE  EQU       1,PB                            INITIAL QUE TYPE
LONGWAIT  EQU       2,PB                            ZOE WAIT FLAG

*                                             LOGIC SWITCHES:
TDRSS     EQU       1,L                             DOWNLINK GATE
DOMSAT    EQU       2,L                             UPLINK GATE
ZONE      EQU       3,L                                   ZONE OF EXCLUSION

*
***** VARIABLE DECLARATIONS
*
          REAL      &MINHOUR                        MINUTES PER HOUR
          REAL      &SECMIN                         SECONDS PER MINUTE
          REAL      &MICRO                          MICROSECONDS PER SECOND
```

```
            REAL        &LENGTH                 SIMULATION LENGTH
            REAL        &TDSTIM                 TDRSS PACKET TIME
            REAL        &DOMTIM                 DOMSAT PACKET TIME
*                                       ZONE OF EXCLUSION:
            REAL        &ZEITMIN            ITERATION:MINIMUM
            REAL        &ZEITAVG                        AVERAGE
            REAL        &ZEITMAX                        MAXIMUM
            REAL        &ZEDURMIN                DURATION:  MINIMUM
            REAL        &ZEDURAVG                           AVERAGE
            REAL        &ZEDURMAX                           MAXIMUM
*                                       HIGH RATE PACKETS:
            REAL        &HRITMIN            ITERATION:MINIMUM
            REAL        &HRITAVG                        AVERAGE
            REAL        &HRITMAX                        MAXIMUM

*
******      CONSTANT DEFINITIONS
*
HOURS       SYN         1                       HOURS OF SIMULATION (IN HOURS)
ERRORCK     SYN         10                      ERROR CHECK TIME (IN MICROSECONDS)

*                                       ZONE OF EXCLUSION:
*                                           ITERATION: (IN MINUTES)
ZEITMN      SYN         75                              MINIMUM
ZEITAV      SYN         90                              AVERAGE
ZEITMX      SYN         150                             MAXIMUM
*                                           DURATION: (IN MINUTES)
ZEDURMN     SYN         0                               MINIMUM
ZEDURAV     SYN         7                               AVERAGE
ZEDURMX     SYN         14                              MAXIMUM

*                                       LOW RATE PACKETS:
*                                           ITERATION: (IN MICROSECONDS)
LRITMIN     SYN         337                             MINIMUM
LRITAVG     SYN         375                             MEAN
LRITMAX     SYN         412                             MAXIMUM

*                                       HIGH RATE PACKETS:
*                                           ITERATION: (IN MINUTES)
HRITMN      SYN         0                               MINIMUM
HRITAV      SYN         5                               MEAN
HRITMX      SYN         8                               MAXIMUM
*                                           DURATION: (IN PACKETS/BURST)
HRDURMIN    SYN         0                               MINIMUM
HRDURAVG    SYN         15200                           MEAN
HRDURMAX    SYN         25000                           MAXIMUM
```

```
*                                   RETRANSMISSION: (IN MICROSECONDS)
RETRYMIN  SYN       27                          MINIMUM
RETRYAVG  SYN       30                          MEAN
RETRYMAX  SYN       35                          MAXIMUM

*
******    VARIABLE INITIALIZATION
*
          LET       &MINHOUR=60.0               MINUTES PER HOUR
          LET       &SECMIN=60.0                SECONDS PER MINUTE
          LET       &MICRO=1.0E6                MICROSECONDS PER SECOND
          LET       &TDSTIM=27.41333            TDRSS PACKET TIME
          LET       &DOMTIM=164.48              DOMSAT PACKET TIME

          LET       &LENGTH=HOURS*&MINHOUR*&SECMIN*&MICRO

*                                       ZONE OF EXCLUSION:
*                                                ITERATION:
          LET       &ZEITMIN=ZEITMN*&SECMIN*&MICRO      MINIMUM
          LET       &ZEITAVG=ZEITAV*&SECMIN*&MICRO      AVERAGE
          LET       &ZEITMAX=ZEITMX*&SECMIN*&MICRO      MAXIMUM
*                                                DURATION:
          LET       &ZEDURMIN=ZEDURMN*&SECMIN*&MICRO        MINIMUM
          LET       &ZEDURAVG=ZEDURAV*&SECMIN*&MICRO        AVERAGE
          LET       &ZEDURMAX=ZEDURMX*&SECMIN*&MICRO        MAXIMUM

*                                       HIGH RATE PACKETS:
*                                                ITERATION:
          LET       &HRITMIN=HRITMN*&SECMIN*&MICRO      MINIMUM
          LET       &HRITAVG=HRITAV*&SECMIN*&MICRO      AVERAGE
          LET       &HRITMAX=HRITMX*&SECMIN*&MICRO      MAXIMUM

*
***** SIMULATION CONTROL
*
          SIMULATE
*
***** TERMINATION TRANSACTION
*
LAST      GENERATE  &LENGTH
          TERMINATE 1

*
***** LOW RATE DATA PACKETS
*
LRBEGIN   GENERATE  RVTRI(200,LRITMIN,LRITAVG,LRITMAX),,,,,_
```

```
                    0PH,0PL,2PB,0PF
          ASSIGN    BIRTHQUE,LRQUE,,PB
          ASSIGN    LONGWAIT,0,,PB
          SPLIT FN$LRDUR,LRWAIT
LRWAIT    QUEUE     PB$BIRTHQUE
          GATE LS   ZONE,LRDONE
          ASSIGN    LONGWAIT,1,,PB
          QUEUE     ZOEWAIT
LRDONE    TRANSFER  ,DOWNLINK

*
***** HIGH RATE DATA PACKETS
*
HRBEGIN   GENERATE  RVTRI(300,&HRITMIN,&HRITAVG,&HRITMAX),,,,1,_
                    0PH,0PL,2PB,0PF
          ASSIGN    BIRTHQUE,HRQUE,,PB
          ASSIGN    LONGWAIT,0,,PB
          SPLIT RVTRI(310,HRDURMIN,HRDURAVG,HRDURMAX),HRWAIT
HRWAIT    QUEUE     PB$BIRTHQUE
          GATE LS   ZONE,HRDONE
          ASSIGN    LONGWAIT,1,,PB
          QUEUE     ZOEWAIT
HRDONE    TRANSFER  ,DOWNLINK

*
***** ZONE OF EXCLUSION
*
ZOE       GENERATE  RVTRI(400,&ZEITMIN,&ZEITAVG,&ZEITMAX),,,,2
          QUEUE     ZOEQUE
          LOGIC S   ZONE
          LOGIC S   TDRSS
          ADVANCE   RVTRI(410,&ZEDURMIN,&ZEDURAVG,&ZEDURMAX),0
          LOGIC R   TDRSS
          LOGIC R   ZONE
          DEPART    ZOEQUE
          TERMINATE 0

*
***** TDRSS DOWNLINK
*
DOWNLINK  QUEUE     TDRSQUE
          GATE LR   TDRSS
          LOGIC S   TDRSS
          DEPART    PB$BIRTHQUE
          TEST E    PB$LONGWAIT,1,SENDDOWN
          DEPART    ZOEWAIT
```

```
SENDDOWN        ADVANCE   &TDSTIM,0
                LOGIC R   TDRSS
                DEPART    TDRSQUE

*
******          DATA INTERFACE FACILITY
*
DIF             QUEUE     DIFQUE
                ADVANCE   ERRORCK,0
                DEPART    DIFQUE
                TRANSFER  .001,UPLINK,RETRY

*
******          REQUEST RETRANSMISSION
*
RETRY           QUEUE     PB$BIRTHQUE
                ADVANCE   RVTRI(500,RETRYMIN,RETRYAVG,RETRYMAX)
                TRANSFER  ,DOWNLINK

*
******          DOMSAT UPLINK
*
UPLINK          QUEUE     DOMQUE
                GATE LR   DOMSAT
                LOGIC S   DOMSAT
                ADVANCE   &DOMTIM,0
                LOGIC R   DOMSAT
                DEPART    DOMQUE
                TERMINATE 0

*
***** CONTROL STATEMENTS
*
                REALLOCATE COM,4000000
                INITIAL   LR$TDRSS
                INITIAL   LR$DOMSAT
                INITIAL   LR$ZONE
                START     1
                END
```

2.4 STORAGE FACILITY (A. K. Petersen)

The vast quantity of data flowing into the DHC will require an extraordinary storage facility to archive it. Such a facility has been identified. The Mass Storage System (MSS), built by the Garland division of E-Systems, Inc. will meet all the requirements. The MSS is essentially a library of high density tapes, located and loaded by computer controlled robotic arms.

The MSS is designed to be modular, expandable, and configurable. Figure 1 shows the floor plan of a basic storage module.

The Data is stored on D-2 high density tape cartridges. Each cartridge has a capacity of approximately 70 GBytes (70×10^9) of information. The tapes are housed in racks, 4 feet wide, 6 feet high and 1 foot deep. A single rack will store 13 TBytes (13×10^{12}) of data. The racks are arranged in pairs, each facing the other.

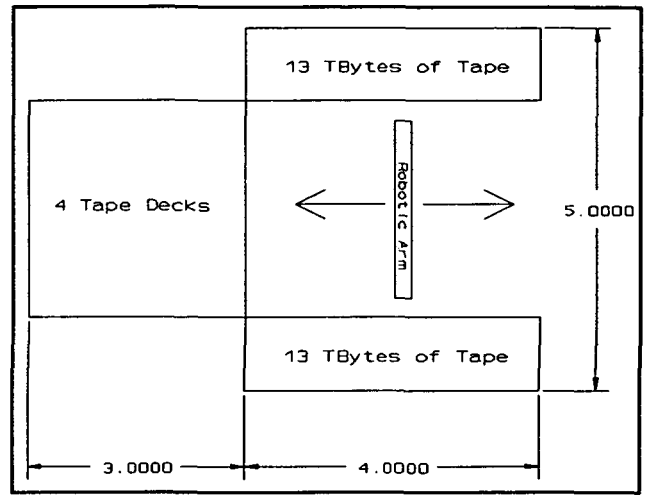


Figure 4 Basic MSS Module

The robotic arm travels between the two racks, selecting tapes from either. Pairs of racks may be adjoined to produce a long aisle, lined on both sides with tapes. As the robotic arm selects tapes, a bar code label on each tape is read by means of a scanner, attached to the arm. In this way, the identity of the tape is not dependent on it's location.

At one end of the aisle, a rack of 4 tape decks is located. The robotic arm loads tapes into these decks for data retrieval. If the stored data is to be modified, the updated information is written back onto the same cartridge. The same decks are loaded with blank tapes for introducing new data into the system. The tape decks are independent of each other in the sense that one can be reading a tape while another is writing. Due to the modular nature of the tape rack, another 4 decks can be located at the other end of the aisle. Figure 2 illustrates a long aisle with decks at each end.

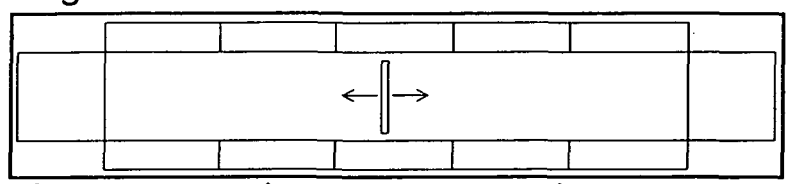


Figure 5 Typical MSS Data Aisle

There is a limit as to how far an aisle can be extended. The retrieval time increases as the mean distance traveled by the robotic arm increases. The optimal capacity-to-speed ratio is achieved with approximately a 48 foot aisle. This is in the neighborhood of 200 TBytes of data. To extend the

capacity of the system beyond this point, and still maintain a reasonable data retrieval time, additional aisles may be constructed.

Building another aisle next to an existing one does not, however, double the capacity of the system. A single row of tape racks exists between the aisles. The tapes in this rack are accessible to the robotic arms in either of the two adjacent aisles. This reduces the potential of a single point failure (robotic arm) rendering a large amount of data inaccessible. Figure 3 illustrates a large MSS with three robotic arms. The maximum capacity offered by the MSS is 10,000 TBytes (10^{16} bytes).

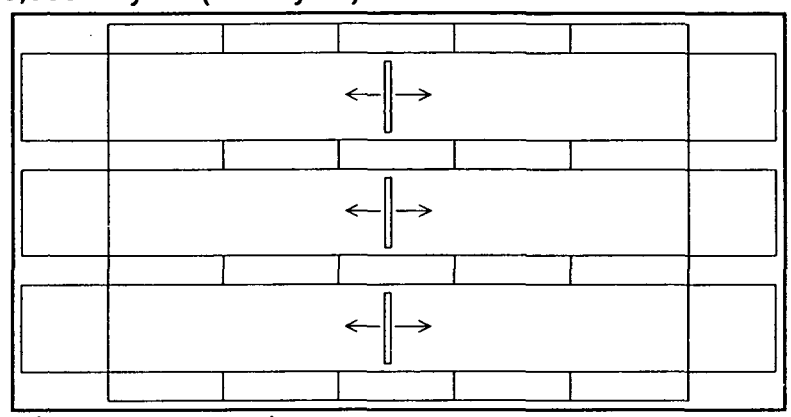


Figure 6 Typical MSS System

Controlling all this storage is an array of mini-super computers. Currently the Convex C-240 is designated as the control unit. Depending on the size of the MSS, more than one computer may be needed. Each computer has 500 GBytes of disk storage attached. This storage acts as a buffer between the tapes and the outside world. The computers are responsible for managing this buffer, implementing the migration algorithms which transfer data to and from tape, and responding to requests for stored data.

Present estimates of the average data access time is on the order of 30 to 45 seconds. This represents the amount of time needed to locate and load the correct tape cartridge, transfer at least some of the information to the disk drives, and make it available to the user. A higher level view is that 30 to 45 seconds may elapse between opening a file and being able to read it.

One of the most important aspects of the MSS is the fact that the entire storage facility (computers, tape decks, robotic arms, and tape racks) appear to the outside world to be a standard Unix device (although a hugh one). Thus no modification of existing software or operating systems need be made.

The modular nature of the system means it can start small and grow as data storage requirements and funding increase. To a great extent, new capacity can be added to the system with out disrupting the existing facility. Modularity also means the performance/cost ratio can be fine tuned by making the aisles longer or shorter, adding more aisles, and adjusting the number of tape decks and computer controllers.

One drawback does exist, the MSS has not yet been built. Smaller prototypes are functioning, but the large system is now only a design. Delivery of the first MSS (to an "unnamed government agency" at Ft. Mead Maryland) is anticipated in the 1992-93 time

frame. This schedule appears to be firm. The MSS facility is the result of a marriage among many technologies, all of which exist today. The controlling computers are available and proven. The tape decks are commercial available, although some modifications are necessary to implement error detection and correction so as to achieve the bit error rate needed. The robotic arms are used in the current prototypes, and the storage media is widely available.

No provisions are currently made for printing the bar code labels attached to each tape cartridge. This does not present a problem, however. The technology for producing such labels using dot matrix printers is widespread. Many industries are using automated label fixing machines, so it would seem this is no more than another instance of integrating existing technologies.

The largest mechanical problem to be addressed is the mechanism by which new, blank tape cartridges are introduced to the system. The current method involves placing the tapes, one at a time, into an import/export cavity. This method is much too slow and labor intensive for the purposed quantity of data. Some sort of mechanical feeder will have to be designed which will accept large shipments of blank tapes, and make then available to the system on demand.

E-Systems has reported no potential problems in meeting all the design criteria of the Space Station Information System (SSIS) storage facility. These criteria include: data rate, error rate, capacity, retrieval rate, and reliability. Cost has yet to be determined, but is estimated in the neighborhood of 2 to 4 millon dollars for the 1400 TBytes required.

Subtask 3

Joseph Pfeiffer, Jr., PhD

1.1. Summary

Subtask three involved two parts: consideration of hardware and software standards applicable to the DHS, and development of benchmarks for evaluating the effectiveness of hardware and software configurations.

In the standards area, three emerging standards were considered: the POSIX standard for Unix implementations (IEEE P1003), in particular its realtime extensions (IEEE P1003.4); the Futurebus+ backplane, and the HiPPI Input/Output architectures. At this time, all three appear to be well-suited to the task at hand.

A suite of benchmarks was developed capable of evaluating a computer system's disk response and communications using TCP. Benchmarks were also obtained from Computer Sciences Corporation for use in evaluating speed of packet dissassembly. Unfortunately, results from the disk and communications benchmarks require substantial interpretation; attempts to evaluate systems without detailed knowledge of the algorithms used for scheduling will not be successful. Also, the CSC packet dissassembly benchmark is flawed in its timings; modifications are required before this benchmark can be considered to give reliable results.

1.2. Standards

An early conclusion in this study was that hardware and software conforming to published standards should be used where possible, i.e. where such conforming configurations have adequate performance. Both vendor-supplied proprietary hardware and software, and NASA-developed hardware and software, should be used only where absolutely necessary. This should result in both cost savings and better long-term support. It was found that conforming hardware and software is available, or is likely to become available, which is of sufficiently high performance.

1.2.1. Software

Three options were originally identified for operating system software: software conforming to POSIX, proprietary vendor software, and a NASA-developed custom operating system. These options were quickly found to collapse to only two, as vendors are quickly conforming their operating systems to POSIX. Even those vendors which use proprietary operating systems (e.g. Digital Equipment Corporation), are offering Unix derivatives as secondary operating systems (in DEC's case, Ultrix) or providing POSIX-compliant interfaces to their operating systems. In other cases (e.g. WindRiver's VxWorks), even though the vendor does not refer to the system as a Unix derivative it is, indeed, POSIX-compliant.

It should be noted that P1003.4, the set of real-time extensions to POSIX, is of most interest here; however, vendors are not willing to commit themselves to this standard until it has been finalized. However, it seems clear that they generally do plan to conform to it at that time.

The DHS was identified as a time-critical application, leading to the following criteria in evaluating operating system implementations: efficiency, bounded response, development environment, and platform independence.

Efficiency

> The selected operating system must impose a minimal load on the system, in order to maximize throughput for a given hardware platform. The efficiency, as it relates to the DHS problem, can be measured by the use of interprocess communication (IPC) and I/O benchmarks, as these two operations will dominate the system time required.

Bounded Response

> More important than raw performance, however, is that the performance be predictable. In order to meet the time-critical repsonse needs of real-time data, it must be possible to bound the response time of all components of the system, including software.
>
> The response of the system may be sensitive to a number of factors, depending on coding of the DHS application. In some cases, operating systems which are not particularly well suited to real time applications may still be useable for this application through identification of factors affecting maximum response time, and careful coding. The same benchmarks used to determine throughput can also be used to determine the distribution of response times.

Development Environment

> The primary costs of the DHS project will not be hardware or off-the-shelf software. Rather, the primary cost will be in GSFC development of software specific to the DHS application. Consequently, reliability can be enhanced and development costs substantially reduced through selection of an operating system which provides a superior development environment.

Platform Independence

> Ideally, the selected operating system should be capable of migrating from one hardware platform to another easily. This requires the availability of the operating system on a variety of platforms. The purpose of this requirement is to reduce the cost of moving the system to a new environment if necessary as a result of developments following initial development.

P1003.4 was evaluated against these criteria, and found to define appropriate control over

nondeterministic events and documented worst-case response times:

Process Priority Heuristics

P1003.4 defines process priority in terms of a set of runnable process lists, with one list for each of (at least) 32 priorities. The head of the highest priority non-empty runnable process is always made a running process.

Priority levels may be either non-preemptive within level (P1003.4 terminology is SCHED_FIFO), preemptive based on a time slice (SCHED_RR, for round robin), or a third policy may be used (SCHED_OTHER). The definition of this third policy, which may function identically to either of the first two or may be completely different, is left up to the implementation. The scheduling policies used by processes at a single level may differ. Even though a process may be running using a non-preemptive scheduling policy, a higher-priority process is capable of preempting it.

A process with appropriate privileges also has the ability to change the priority or policy of another process (or of itself). A process may also voluntarily yield the CPU.

Page Faults

Processes are able to lock specified memory regions in physical memory. Either a specified region, the entire program, the program's data segment, the program's stack segment, or the program's text segment, may be locked. In all cases other than region locking, subsequent growth in the specified segment will also be locked in memory.

Disk Priorities

While P1003.4 does not address the problem of disk priorities directly, a concept of real-time files does exist. This permits a process much greater control over disk accesses than is available in "standard Unix." Some of the items of a real-time file which the program can control include (among others) the initial size of the file when allocated; the amount by which the file is extended when its end is overrun; and whether or not to cache writes. Improved asynchronous IO is also provided by P1003.4; it is possible to read and write to disk files in parallel with other processing. An asynchronous event is raised when the IO operation is completed.

Documented Worst-Case Response Time

P1003.4 requires documented worst-case response times, making possible the accurate estimation of necessary buffer sizes and latency within the DHS. The models of interprocess and interprocessor communication available with Unix

were also found to be acceptable, and usable across several platforms.

Several vendors' (Alliant, Aptec, Convex, Cray, MODCOMP and Sequent) implementations were also evaluated in the light of the above criteria. It was found that, perhaps universally across the industry, vendors' environments either currently are, or soon will be, virtually identical. These vendors' implementations were all found to be acceptable.

In the light of these findings, it is strongly recommended that the DHS adopt a POSIX-compliant operating system. There is no reason to adopt a custom operating system.

1.2.2. Hardware

Two standards, and one commercial product, were examined. The standards include the Futurebus+ (IEEE P896.1) backplane and the High Performance Parallel Interface (HiPPI) input/output architecture. The commerical product was Ultra Technologies' UltraNet.

1.2.2.1. Futurebus+

Futurebus+ is a technology-independent backplane standard supporting packet transfer, permitting estimated bandwidths in excess of 1000MB/s. Unfortunately, vendor support is, as yet, lacking. The vendors whose hardware appears most promising (Aptec, Convex) are based on proprietary busses. When actual product evaluations for purposes of constructing the DHS are performed, however, presence of Futurebus+ should be regarded as a positive factor.

1.2.2.2. HiPPI

The HiPPI I/O standard supports transfers of up to 1600 Mbs, which is adequate for the DHS. Unlike Futurebus+, HiPPI is already well supported by vendors. HiPPI should be adopted as a requirement for hardware to be used in the DHS.

1.2.2.3. UltraNet

Ultra Network Technologies' UltraNet is an example of new communications technology which is becoming available. UltraNet supports reliable network communications using HiPPI I/O at rates of up to one gigabit/sec, which is near the rates required for the DHS. It supports a TCP software interface, with handling of TCP packets performed within the network interface. Also of interest is that it is intended to support connections, rather than connectionless sockets; in contrast to many network technologies currently available, UltraNet actually provides higher transmission rates for reliable than for unreliable communications. It is to be expected that similar products, supporting rates sufficient for the DHS, will become available within the next few years.

1.3. Benchmarks

Benchmark programs have been developed for measuring disk IO and interprocess communi-

cation rates using the Berkeley socket abstraction. Both benchmark sets are capable of doing automated data acquisition and of separately measuring the elapsed time, user time, and system time required for transfers.

1.3.1. Disk IO

The disk IO benchmark measures read and write times to disk. The user is able to specify the size and number of transfers; additional processing can be simulated through iterating a null loop. The system load due to the input or output not accounted for in the benchmark itself is determined by running a separate, low priority process, and measuring the CPU time it is able to acquire.

Interpretation of the benchmark results is made difficult by the use of asynchronous disk transfers, and disk buffers, in the operating system. The first problem to arise is that, in order to operate efficiently, the operating system does not physically write the disk in response to write requests. Instead, the requests are scheduled and performed in some unspecified order. Benchmark results have been found to correspond more directly to these operations than to the actual requests. While it is possible to specify that the requests occur immediately, this ties the benchmark results to the speed of the disk medium, which produces more serious distortions than those from the disk scheduling. While not insurmountable, this does require a significant amount of interpretation. In order to facilitate this interpretation, the disk IO benchmark provides an accounting of the number of actual physical transfers to be made in the course of the benchmark execution. Table I shows the form of the output from the benchmark. The sample is from a benchmark run on a Sun Sparcstation1+, giving write times. The results are in the form of a pair of lines for each benchmark.

Table I

1.190000 1.050000 0.080000 1000 1024 1.760000 0.020000 0.520000 126 1
3.430002 2.320000 0.230000 1000 2048 3.510000 0.040000 0.860000 251 0
5.090001 3.480000 0.390000 1000 3072 5.130000 0.020000 1.240000 376 1
6.640002 4.540000 0.480000 1000 4096 6.740000 0.020000 1.670000 501 1
12.850002 9.870000 0.730000 1000 5120 12.920000 0.020000 2.140000 626 157
12.550000 9.260000 0.940000 1000 6144 12.590000 0.020000 2.320000 751 97
17.900001 14.000000 1.080000 1000 7168 17.920000 0.000000 2.830000 876 235
13.250002 9.830000 0.970000 1000 8192 13.920000 0.030000 3.010000 1001 0

The two lines are sometimes reversed, as they are written by two separate processes. The parent line gives (with the numbers from the first run from Table I inserted as an example):

1. The number of transfers requested (1000)

2. The size of each requested transfer (1024 bytes)

3. The elapsed time for the benchmark (1.76 seconds)

4. The user time taken by the benchmark (0.02 seconds)

5. The system time taken by the benchmark (0.52 seconds)

6. The number of blocks written to disk (126)

7. The number of blocks read from disk (1)

The child line gives statistics for the child process used to determine the amount of unaccounted time used. Its fields are:

1. The elapsed time for the child (1.19 seconds)

2. The user time taken by the child (1.05 seconds)

3. The system time taken by the child (0.08 seconds)

In this example, a total of 0.09 seconds of CPU time is not accounted for (using the child elapsed time measure). This serves to give a worst-case estimate of the unaccounted overhead. The effect of the scheduling of the disk is clearly seen, for example, in the difference between the time required for the size 7168 buffer (elapsed time 17.92 seconds) compared to that for the size 8192 buffer (elapsed time 13.92 seconds). It should come as no surprise that this system performs all disk reads and writes in 8192 byte blocks.

A second problem to arise is in the use of disk buffers. It has been found that in writing, and then reading, a file there is a possibility that the desired information may already be in the disk buffers, eliminating the need for a physical disk read. This problem is avoided by ensuring that the files to be read or written must be larger than the disk buffers. The effect can be seen in Table II, which gives the read times for the same benchmark run.

Table II

0.600000 0.300000 0.000000 1000 1024 0.820000 0.030000 0.410000 1 0
1.210002 0.700000 0.000000 1000 2048 1.350000 0.030000 0.650000 1 0
1.660003 0.890000 0.000000 1000 3072 1.810000 0.020000 0.920000 1 1
2.470000 1.160000 0.000000 1000 4096 2.570000 0.010000 1.310000 1 1
3.300000 1.670000 0.040000 1000 5120 3.280000 0.020000 1.630000 1 66
3.320000 1.730000 0.020000 1000 6144 3.570000 0.010000 1.780000 1 16
4.850002 2.450000 0.180000 1000 7168 4.930000 0.040000 2.270000 1 154
7.410002 4.350000 0.340000 1000 8192 7.430000 0.040000 2.570000 1 322

Essentially no reads are required until the size of the file exceeds four megabytes, as the data is already in the disk buffers due to previous runs.

1.3.2. Interprocess Communications

A second benchmark measures interprocess communications using TCP and either reliable (SOCK_STREAM) or unreliable (SOCK_DGRAM) protocols. As before, the user is able to specify packet size and number of packets to transfer; additional processing can be simulated with a null loop.

The benchmark is run on two hosts connected by a network, with separate measurements taken of the sending and the receiving host. When an unreliable protocol is tested, a complication arises in that the number of packets sent and the number received may differ. In order to obtain information regarding both the sender and the receiver, two runs are taken. In order to obtain information about the sender and the number of failing packets, a run is made in which the receiver attempts to receive as many packets as were sent; the receiver will generally fail to receive all the packets and will time out in this run. In a second run, the sender attempts to send twice as many packets as will be received; in this run, information is obtained regarding the receiver.

Table III shows the form of the output from the sender data collection. The results are from a run using two Sun 3/60 workstations, using an unreliable protocol.

Table III

Thu Jan 23 22:13:57 MST 1992
zia sender data
zia sender: 10000 1 0: 11419 360 10900
zia sender: 10000 256 0: 19140 360 15980
zia sender: 10000 512 0: 15499 820 14480
zia sender: 10000 768 0: 17080 740 16080
zia sender: 10000 1024 0: 18680 780 17700
zia sender: 10000 1280 0: 24860 680 23920
zia sender: 10000 1536 0: 30700 800 29660
zia sender: 10000 1792 0: 38339 880 31520
zia sender: 10000 2048 0: 36120 920 33660
zia sender: 10000 2304 0: 45739 460 44140
zia sender: 10000 2560 0: 45340 700 41040
zia sender: 10000 2816 0: 49140 740 42340
zia sender: 10000 3072 0: 52180 780 50660
zia sender: 10000 3328 0: 78640 680 61420
zia sender: 10000 3584 0: 61900 320 58640
zia sender: 10000 3840 0: 63980 400 60360
zia sender: 10000 4096 0: 65240 720 61440

The columns give (using the results from the first row):

1. The number of packets sent (10000)

2. The size of a packet (1 byte)

3. The number of null loop iterations between transfers (0)

4. The elapsed time for the benchmark (11419 milliseconds)

5. The user time (360 milliseconds)

6. The system time (10900 milliseconds)

A surprise in these results is the close correlation between the elapsed time and the system time.

Table IV shows the form of the receiver data, for the second (receiver data) run. This example is shown before the receiver data from the first run, in order to show the "normal" form of the data.

Table IV

Sat Jan 25 04:14:36 MST 1992
zia receiver data

puye receiver:	10000	1	0: 23260 260 4320; 0
puye receiver:	10000	256	0: 16539 180 6740; 0
puye receiver:	10000	512	0: 15780 460 7860; 0
puye receiver:	10000	768	0: 17439 480 8600; 0
puye receiver:	10000	1024	0: 19000 400 7440; 0
puye receiver:	10000	1280	0: 25120 240 7180; 0
puye receiver:	10000	1536	0: 30980 380 8100; 0
puye receiver:	10000	1792	0: 32819 520 11120; 0
puye receiver:	10000	2048	0: 35460 320 11040; 0
puye receiver:	10000	2304	0: 45479 340 11180; 0
puye receiver:	10000	2560	0: 42559 200 10520; 0
puye receiver:	10000	2816	0: 44079 480 11340; 0
puye receiver:	10000	3072	0: 52759 360 13780; 0
puye receiver:	10000	3328	0: 62059 380 13300; 0
puye receiver:	10000	3584	0: 59339 400 13960; 0
puye receiver:	10000	3840	0: 61100 260 14960; 0
puye receiver:	10000	4096	0: 62580 280 14900; 0

Here, the first six columns correspond to those in the sender data. The seventh column gives the number of packets which required more than one read to obtain the complete packet. When using a reliable protocol, this number is typically large; in some cases, we have seen nearly all packets arrive broken (this is not a typographical error: using the unreliable protocol, packets are occasionally lost altogether, but we rarely see a broken one. Using the reliable protocol, while all data does arrive, packets very frequently arrive in pieces).

Table V shows the results of the first run, for the receiver. As can be seen, greater than 1%

of the packets sent may typically be lost.

Table V

Fri Jan 24 22:47:11 MST 1992
zia sender data

puye receiver:	10000	1	0:	11220 260 4140; 0
puye receiver:	10000	256	0:	timed out; received 9949 packets
puye receiver:	10000	512	0:	15299 400 7540; 0
puye receiver:	10000	768	0:	timed out; received 9938 packets
puye receiver:	10000	1024	0:	timed out; received 9865 packets
puye receiver:	10000	1280	0:	timed out; received 9923 packets
puye receiver:	10000	1536	0:	timed out; received 9929 packets
puye receiver:	10000	1792	0:	timed out; received 9918 packets
puye receiver:	10000	2048	0:	timed out; received 9920 packets
puye receiver:	10000	2304	0:	timed out; received 9904 packets
puye receiver:	10000	2560	0:	timed out; received 9919 packets
puye receiver:	10000	2816	0:	timed out; received 9945 packets
puye receiver:	10000	3072	0:	timed out; received 9922 packets
puye receiver:	10000	3328	0:	timed out; received 9939 packets
puye receiver:	10000	3584	0:	timed out; received 9937 packets
puye receiver:	10000	3840	0:	timed out; received 9933 packets
puye receiver:	10000	4096	0:	timed out; received 9948 packets

It should be noted that in the receiver data presented here, the system time is substantially lower than the elapsed time. This is a more expected result.